

DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUU	UUU	GGGGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUU	UUU	GGGGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUU	UUU	GGGGGGGGGGGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG

[illegible]

```
1 0001 0 MODULE DBGEVALOP (IDENT = 'V04-000') =
2 0002 0
3 0003 1 BEGIN
4 0004 1
5 0005 1 *****
6 0006 1 *
7 0007 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
8 0008 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
9 0009 1 * ALL RIGHTS RESERVED. *
10 0010 1 *
11 0011 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
12 0012 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
13 0013 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
14 0014 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
15 0015 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
16 0016 1 * TRANSFERRED. *
17 0017 1 *
18 0018 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
19 0019 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
20 0020 1 * CORPORATION. *
21 0021 1 *
22 0022 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
23 0023 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
24 0024 1 *
25 0025 1 *****
26 0026 1
27 0027 1
28 0028 1 WRITTEN BY
29 0029 1 R. Title May, 1982
30 0030 1 P. Sager Jun, 1982
31 0031 1
32 0032 1 MODULE FUNCTION
33 0033 1 The top level routine in this module, DBG$EVAL_LANG_OPERATOR,
34 0034 1 is called by the expression parser to perform an operation.
35 0035 1 This routine does type conversion, as specified by language
36 0036 1 specific type conversion tables, if needed. It then calls
37 0037 1 the routine specified by language specific operator routine
38 0038 1 tables, in order to actually perform the operation.
39 0039 1
40 0040 1
41 0041 1 REQUIRE 'SRC$:DBGPROLOG.REQ';
42 0175 1
43 0176 1 LINKAGE
44 0177 1 PLI_LINK = CALL(REGISTER=0, REGISTER=1, REGISTER=2, REGISTER=3);
45 0178 1
46 0179 1 FORWARD ROUTINE
47 0180 1 AAA DUMMY,
48 0181 1 DBG$BLISS_BITSELECT: NOVALUE,
49 0182 1 DBG$BLISS_INDIRECTION,
50 0183 1 DBG$CONV_TEXT_VALUE,
51 0184 1 DBG$CONV_TQUADWORD_VALUE,
52 0185 1 DBG$CONV_TOCTAWORD_VALUE,
53 0186 1 DBG$CONV_TRFA_VALUE,
54 0187 1 DBG$DO_MAPPING: NOVALUE,
55 0188 1 DBG$EVALOP_SET_LANGUAGE: NOVALUE,
56 0189 1
57 0190 1 DBG$EVAL_LANG_OPERATOR,
```

! Dummy routine to force dump of tables
! Perform BLISS bit selection X<p,s,e>
! Performs BLISS indirection
! Convert text string to value
! Convert text string to QUADWORD
! Convert text string to OCTAWORD
! Convert text string to RFA value
! Perform type mapping
! Set up pointer to Operator Information
! Tables
! Evaluate a language expr. operator

58	0191	1	DBG\$GET_SET_TYPEID,	Set the parent TYPEID for SET data type
59	0192	1	DBG\$GET_DTYPE,	Obtain dtype from Value Descriptor
60	0193	1	DBG\$LANGUAGE_TYPE_CONV,	Language-specific type converter
61	0194	1	DBG\$MAP_DTYPE_CLASS,	Given DTYPE, do a best guess at the CLASS
62	0195	1		
63	0196	1	DBG\$NUM_BYTES,	Returns number of bytes to hold a given DTYPE.
64	0197	1		
65	0198	1	DBG\$PERFORM_TYPEID_CHECK,	Perform typeid check on non-atomic data types
66	0199	1		
67	0200	1	DBG\$TYPE_CONV,	Top level DEBUG type converter
68	0201	1	CONV_TEXT_PACK_VALUE,	A routine convert the unconverted string packed decimal
69	0202	1		
70	0203	1	GET_DATA_LENGTH,	Get the length of the data
71	0204	1	GET_SCALE,	Get scaling factor for Floating Point data to packed decimal conversion
72	0205	1		
73	0206	1	FIND_JOIN,	Find join in directed acyclic graph
74	0207	1	FIND_PATH,	Find path in directed acyclic graph
75	0208	1	FIND_PATH_DEPOSIT,	Similar to FIND_PATH, used for DEPOSIT
76	0209	1	FIXUP_EMPTY_SET,	Fix up typeid for empty set
77	0210	1	INTMED_DATA_FOR_DEP,	Create intermediate data for deposit
78	0211	1	MAKE_VAL_DESC,	Build a dummy value descriptor
79	0212	1	MAP_NRO_DTYPE_IN_RPG,	Map normal and alternate least significant digit and sign into RPG standard output format
80	0213	1		
81	0214	1		
82	0215	1	MAP_PACKED: NOVALUE,	Maps packed decimal to correct type.
83	0216	1	MAP_PLI_TYPE_SIZE: NOVALUE,	Maps dtype to PL/I specific type; calculates size.
84	0217	1	MODIFY_PLI_TARGET_TYPE: NOVALUE,	Modify PLI target type
85	0218	1	PLI_TYPE_CONV,	PLI Type Conv. used for PLI bit-string conversion
86	0219	1		
87	0220	1	PLI_HANDLER,	Catches PL/I conversion errors.
88	0221	1	TYPEID_CHECK_ENUM,	Perform typeid check on a pair of enum.
89	0222	1	TYPEID_CHECK_SET,	Perform typeid check on a pair of sets
90	0223	1	TYPEID_RANGE_CHECK_ENUM,	Perform range check for enum data type
91	0224	1	TYPEID_RANGE_CHECK_SUBRNG;	Perform range check for subrange data type
92	0225	1		
93	0226	1	EXTERNAL ROUTINE	
94	0227	1	FOR\$CVT_D_TE,	Fortran E format routine
95	0228	1	FOR\$CVT_G_TE,	Fortran E format routine
96	0229	1	FOR\$CVT_H_TE,	Fortran E format routine
97	0230	1	MTH\$JNOT,	Bitwise complement of a Longword
98	0231	1	OT\$SCVT_FB_L,	Convert text Binary to Longword
99	0232	1	OT\$SCVT_TI_L,	Convert text (signed) to Longword
100	0233	1	OT\$SCVT_T_F,	Convert text to single Floating
101	0234	1	OT\$SCVT_T_D,	Convert text to Double Float
102	0235	1	OT\$SCVT_T_G,	Convert text to G_Float
103	0236	1	OT\$SCVT_T_H,	Convert text to H_Float
104	0237	1	OT\$SCVT_TO_L,	Convert text (octal) to Longword
105	0238	1	OT\$SCVT_TZ_L,	Convert text (hexadecimal) to Longword
106	0239	1	PLISCHARABIT_R6: PLI_LINK,	Convert text to aligned bit-string.
107	0240	1	PLISCVT_ANY,	Used in PL/I bit-strings conversions.
108	0241	1	DBG\$COLLECT: NOVALUE,	"Sanitize" Primary Descriptors
109	0242	1	DBG\$COVER_DX_DX,	Debug type converter.
110	0243	1	DBG\$CVT_DX_DX: NOVALUE,	Debug version of LIB\$CVT_DX_DX
111	0244	1	DBG\$CVT_TQUADWORD TO VALUE,	Convert into QUADWORD value
112	0245	1	DBG\$CVT_TUQUADWORD TO VALUE,	Convert into unsigned QUADWORD value
113	0246	1	DBG\$CVT_TOCTAWORD TO VALUE,	Convert into OCTAWORD value
114	0247	1	DBG\$CVT_TRFA_TO_VALUE,	Convert into RFA value

```
115 0248 1 DBGSGET_DST_NAME,      ! Obtain name of symbol
116 0249 1 DBGSGET_TEMPHEM,      ! Allocate temporary memory
117 0250 1 DBG$MAKE_SKELETON_DESC, ! Create skeleton Value Descriptor
118 0251 1 DBG$MAKE_VAL_DESC,     ! Convert VMS desc to valdesc
119 0252 1 DBG$MAKE_VMS_DESC,     ! Convert Primary to VMS desc
120 0253 1 DBG$NEWLINE: NOVALUE,  ! Close print line and start new line
121 0254 1 DBG$PERFORM_OPERATOR: NOVALUE, ! Perform arithmetic operation
122 0255 1 DBG$PRIM_TO_ADDR,      ! Convert primary descriptor to value
123 0256 1                        ! descriptor containing the address
124 0257 1                        ! of the primary.
125 0258 1 DBG$PRIM_TO_VAL,       ! Converts a primary descriptor
126 0259 1                        ! to a value descriptor
127 0260 1 DBG$PRINT: NOVALUE,    ! Print FA0-formatted text
128 0261 1 DBG$STA_SYMVALUE: NOVALUE, ! Get symbol's value
129 0262 1 DBG$STA_TYP_ATOMIC: NOVALUE, ! Get symbol information
130 0263 1 DBG$STA_TYP_ENUM: NOVALUE, ! Get symbol information
131 0264 1 DBG$STA_TYP_PICT: NOVALUE, ! Get symbol information
132 0265 1 DBG$STA_TYP_SET: NOVALUE, ! Get symbol information
133 0266 1 DBG$STA_TYP_SUBRNG: NOVALUE, ! Get symbol information
134 0267 1 DBG$STA_TYP_TYPEDPTR: NOVALUE, ! Get symbol information
135 0268 1 DBG$TYPEID_FOR_SET;    ! Construct a Set Constant typeid
136 0269 1
137 0270 1 EXTERNAL
138 0271 1 DBG$GB_LANGUAGE: BYTE,  ! Current lang setting
139 0272 1 DBG$GL_NEG_CONST_TOKEN, ! Negative constant token
140 0273 1 DBG$GL_POS_CONST_TOKEN, ! Positive constant token
141 0274 1 DBG$GL_NEG_SIGN_TOKEN,  ! Negative constant token (--> unary minus)
142 0275 1 DBG$GL_POS_SIGN_TOKEN,  ! Positive constant token (--> unary plus)
143 0276 1 DBG$GL_DEVELOPER: BITVECTOR[]; ! Set developer flag
144 0277 1
145 0278 1 BUILTIN
146 0279 1 ASHP,
147 0280 1 EDITPC;
148 0281 1
149 0282 1 LITERAL
150 0283 1 DBG$K_DTYPE_ARRAY = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_ARRAY,
151 0284 1 DBG$K_DTYPE_ENUM = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_ENUM,
152 0285 1 DBG$K_DTYPE_TPTR = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_TPTR,
153 0286 1 DBG$K_DTYPE_SET = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_SET,
154 0287 1 DBG$K_DTYPE_SUBRNG = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_SUBRNG,
155 0288 1 DBG$K_DTYPE_PTR = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_PTR,
156 0289 1 DBG$K_DTYPE_PICT = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_PICT,
157 0290 1 DBG$K_DTYPE_RFA = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_RFA;
158 0291 1
159 0292 1
160 0293 1 ! The following is so that the map table can do a mapping from
161 0294 1 ! DST$K_BOOL to DSC$K_DTYPE_TF
162 0295 1
163 0296 1 LITERAL
164 0297 1 DSC$K_DTYPE_BOOL = DST$K_BOOL;
165 0298 1
166 0299 1
167 0300 1 ! The following is so that the lang. cvt table can do a conversion from
168 0301 1 ! DST$K_DTYPE_ANY to a specify dtype or a specify dtype to DST$K_DTYPE_ANY.
169 0302 1
170 0303 1 LITERAL
171 0304 1 DSC$K_DTYPE_ANY = %X'FF';
```

```
172 0305 1
173 0306 1
174 0307 1 BIND
175 0308 1 TABLEBASE = UPLIT BYTE (%ASCII 'BASE');
176 0309 1
177 0310 1 ! OPCODE_NAME is global so that the type converter can get at it for
178 0311 1 ! purposes of signalling error messages.
179 0312 1
180 0313 1 GLOBAL
181 0314 1 DBG$GL_OPCODE_NAME: REF VECTOR [,BYTE]; ! Operator name in ASCII
182 0315 1
183 0316 1 OWN
184 0317 1 BLISS_BITSELECTION_FLAG1:INITIAL(0); ! Flag saying we've done a bit select
185 0318 1 ! operator in BLISS
186 0319 1 BLISS_BITSELECTION_FLAG2:INITIAL(0); ! Flag saying we've done a bit select
187 0320 1 ! operator in BLISS
188 0321 1 BLISS_INDIRECTION_FLAG, ! Flag saying whether we've done an
189 0322 1 ! indirection yet
190 0323 1 CVT_ROUND_FLAG, ! A flag set to TRUE indicate the
191 0324 1 ! conversion result is rounded
192 0325 1 CVT_TBL: REF VECTOR [,LONG], ! Pointer to a Language Dependent Type
193 0326 1 ! Conversion Table
194 0327 1 CVT_TBL_SIZE, ! Number of entries in CVT_TBL
195 0328 1 CVTINFO_TABLE: REF CVTINFO_TABLE, ! Pointer to the Type Conversion
196 0329 1 ! Table for the current language
197 0330 1 MAP_TBL: REF VECTOR [,WORD], ! Pointer to a Type Mapping Table
198 0331 1 MAP_TBL_SIZE, ! Number of entries in MAP_TBL
199 0332 1 MAX_DEPTH, ! Maximum search depth for graphs
200 0333 1 OPINFO_TABLE: REF OPINFO_TABLE, ! Pointer to the Operator Information
201 0334 1 ! Table for the current language
202 0335 1 SAVE_RESULT_DESC: DBG$STG_DESC; ! Saved descriptor for communication
203 0336 1 ! between <> and . in BLISS
```

..	205	0337	1	!
...	206	0338	1	!
...	207	0339	1	!
...	208	0340	1	!
...	209	0341	1	!
...	210	0342	1	!
...	211	0343	1	!
...	212	0344	1	!
...	213	0345	1	!
...	214	0346	1	!
...	215	0347	1	!
...	216	0348	1	!
...	217	0349	1	!
..	218	0350	1	!

## MACROS FOR OPERATOR ROUTINE TABLES

These macros are used to generate the tables which are used to select operator routines based on the operand data types and which define any type conversion behavior appropriate to the operator. For example, INT\*REAL has to be converted to REAL\*REAL in most languages, after which the REAL\*REAL multiply routine is invoked; this kind of behavior is specified in the tables whose structure is defined here.

```
220 0351 1 OPERATOR ROUTINE TABLE
221 0352 1
222 0353 1
223 0354 1 The Operator Routine Table for a given operator contains one entry for each
224 0355 1 routine which can be associated with that operator. Each such entry contains
225 0356 1 the operand types accepted by that routine (e.g., REAL,REAL for F Floating
226 0357 1 multiply) and a routine index which identifies the routine to be Invoked.
227 0358 1 And also there is an optional routine index for typeid check.
228 0359 1 The types are represented as standard Dtypes and
229 0360 1 the routine index is simply a CASE index used to select the routine body.
230 0361 1
231 0362 1 An Operator Routine Table for an operator is declared as follows:
232 0363 1
233 0364 1 OPERATOR_ROUTINE_TABLE(TBLNAME,
234 0365 1 OPERATOR_ROUTINE(LEFT_TYPE, RIGHT_TYPE, RESULT_TYPE, ROUT_NAME,
235 0366 1 TYPEID_CHECK_ROUT_NAME),
236 0367 1
237 0368 1 OPERATOR_ROUTINE(LEFT_TYPE, RIGHT_TYPE, RESULT_TYPE, ROUT_NAME,
238 0369 1 TYPEID_CHECK_ROUT_NAME));
239 0370 1
240 0371 1 Here TBLNAME is the name of the table. LEFT_TYPE is the Type Index Value of
241 0372 1 the left argument (or the only argument if the operator is unary), RIGHT_TYPE
242 0373 1 is the Type Index Value of the right argument (or 0 if the operator is unary),
243 0374 1 and ROUT_NAME is the name of the corresponding semantic routine. ROUT_NAME
244 0375 1 is automatically prefixed by 'ORTSK_' to yield the routine index value.
245 0376 1 TYPEID_CHECK_ROUT_NAME is the name of the corresponding typeid check semantic
246 0377 1 routine. TYPEID_CHECK_ROUT_NAME is automatically prefixed by 'ORTSK_TYPEID_'
247 0378 1 to yield the routine index value.
248 0379 1
249 0380 1 Define the macros that generate the Operator Routine Table.
250 0381 1
251 0382 1 MACRO
252 M 0383 1 OPERATOR_ROUTINE_TABLE(TBLNAME) =
253 0384 1 BIND TBLNAME = PLIT BYTE(%REMAINING): VECTOR[.BYTE] %;
254 0385 1
255 0386 1 MACRO
256 0387 1 OPERATOR_ROUTINE(LEFT_TYPE, RIGHT_TYPE, RESULT_TYPE, ROUT_NAME,
257 M 0388 1 TYPEID_CHECK_ROUT_NAME) =
258 M 0389 1 XIF %DECLARED (%NAME ('RSTSK_TYPE_', RIGHT_TYPE))
259 M 0390 1 %THEN
260 M 0391 1 DBGSK_MAXIMUM_DTYPE + %NAME ('RSTSK_TYPE_', RIGHT_TYPE)
261 M 0392 1 %ELSE
262 M 0393 1 %NAME ('DSCSK_DTYPE_', RIGHT_TYPE)
263 M 0394 1 %FI
264 M 0395 1 XIF %DECLARED (%NAME ('RSTSK_TYPE_', LEFT_TYPE))
265 M 0396 1 %THEN
266 M 0397 1 DBGSK_MAXIMUM_DTYPE + %NAME ('RSTSK_TYPE_', LEFT_TYPE)
267 M 0398 1 %ELSE
268 M 0399 1 %NAME ('DSCSK_DTYPE_', LEFT_TYPE)
269 M 0400 1 %FI
270 M 0401 1 %NAME ('ORTSK_', ROUT_NAME) AND %X'00FF',
271 M 0402 1 %NAME ('ORTSK_', ROUT_NAME)^-8,
272 M 0403 1 XIF %LENGTH LEQ 4
273 M 0404 1 %THEN
274 M 0405 1 0
275 M 0406 1 %ELSE
276 M 0407 1 %NAME ('ORTSK_TYPEID_', TYPEID_CHECK_ROUT_NAME) AND %X'00FF'
```

```

277      M 0408 1      XFI,
278      M 0409 1      XIF XLENGTH LEQ 4
279      M 0410 1      XTHEN
280      M 0411 1      0
281      M 0412 1      XELSE
282      M 0413 1      XNAME ('ORTSK_TYPEID_', TYPEID_CHECK_ROUT_NAME)^-8
283      M 0414 1      XFI,
284      M 0415 1      XIF XDECLARED (XNAME ('ORTSK_RESULT_', RESULT_TYPE))
285      M 0416 1      XTHEN
286      M 0417 1      XNAME ('ORTSK_RESULT_', RESULT_TYPE)
287      M 0418 1      XELSE
288      M 0419 1      (XIF XDECLARED (XNAME ('RSTSK_TYPE_', RESULT_TYPE))
289      M 0420 1      XTHEN
290      M 0421 1      DBGSK_MAXIMUM_DTYPE + XNAME ('RSTSK_TYPE_', RESULT_TYPE)
291      M 0422 1      XELSE
292      M 0423 1      XNAME ('DSCSK_DTYPE_', RESULT_TYPE)
293      M 0424 1      XFI)
294      M 0425 1      XFI,
295      M 0426 1      0X;
296      M 0427 1
297      M 0428 1
298      M 0429 1      ! The TYPE_GRAPH_EDGE macro is used by the tables below.
299      M 0430 1      !
300      M 0431 1      MACRO
301      M 0432 1      TYPE_GRAPH_EDGE (LOWER_TYPE, HIGHER_TYPE) =
302      M 0433 1      (XIF XDECLARED (XNAME ('RSTSK_TYPE_', HIGHER_TYPE))
303      M 0434 1      XTHEN
304      M 0435 1      DBGSK_MAXIMUM_DTYPE + XNAME ('RSTSK_TYPE_', HIGHER_TYPE)
305      M 0436 1      XELSE
306      M 0437 1      XNAME ('DSCSK_DTYPE_', HIGHER_TYPE)
307      M 0438 1      XFI)^8
308      M 0439 1      OR
309      M 0440 1      (XIF XDECLARED (XNAME ('RSTSK_TYPE_', LOWER_TYPE))
310      M 0441 1      XTHEN
311      M 0442 1      DBGSK_MAXIMUM_DTYPE + XNAME ('RSTSK_TYPE_', LOWER_TYPE)
312      M 0443 1      XELSE
313      M 0444 1      XNAME ('DSCSK_DTYPE_', LOWER_TYPE)
314      M 0445 1      XFI) X;

```

```

316 0446 1 TYPE HIERARCHY TABLE
317 0447 1
318 0448 1 The Type Hierarchy Table specifies the implicit type conversions to
319 0449 1 be done within an expression. A type hierarchy table may be specific
320 0450 1 to a given operator in a given language, or may be shared across
321 0451 1 operators and languages.
322 0452 1
323 0453 1 The Type Hierarchy Table is a directed acyclic graph. For example,
324 0454 1 for the addition operator in Fortran, a part of the Type Hierarchy
325 0455 1 Table might look like this:
326 0456 1
327 0457 1
328 0458 1
329 0459 1
330 0460 1
331 0461 1
332 0462 1
333 0463 1
334 0464 1
335 0465 1
336 0466 1
337 0467 1
338 0468 1
339 0469 1
340 0470 1
341 0471 1
342 0472 1
343 0473 1
344 0474 1
345 0475 1
346 0476 1
347 0477 1
348 0478 1
349 0479 1
350 0480 1
351 0481 1

```

```

0465 1 A Type Hierarchy Table is specified by giving its edges.
0466 1
0467 1 An example of a Type Hierarchy Table definition is:
0468 1
0469 1 TYPE_HIERARCHY_TABLE ( FORADD_TYPE_HIER_TABLE,
0470 1     TYPE_GRAPH_EDGE (L, F),
0471 1     TYPE_GRAPH_EDGE (F, FC),
0472 1     TYPE_GRAPH_EDGE (F, D),
0473 1     ...., 0 );
0474 1
0475 1
0476 1 Define the macro which declares a Type Hierarchy Table.
0477 1
0478 1 MACRO
0479 1     TYPE_HIERARCHY_TABLE (TBLNAME) =
0480 1     BIND TBLNAME = PLIT WORD(%REMAINING) : VECTOR [,WORD] %;
0481 1

```

```
0482 1 TYPE INCOMPATIBILITY TABLE
0483 1
0484 1 The Type Incompatibility Table specifies which pairs of types are
0485 1 incompatible (i.e., it is illegal for both to be in the same
0486 1 operator expression). For example, D FLOAT and G FLOAT are
0487 1 incompatible in FORTRAN. A Type Incompatibility Table may be specific
0488 1 to a given operator in a given language, or may be shared across
0489 1 operators and languages.
0490 1
0491 1 A Type Incompatibility Table is specified by giving a list of
0492 1 incompatible type pairs.
0493 1
0494 1 An example of a Type Incompatibility Table definition is:
0495 1
0496 1 TYPE_INCOMP TABLE ( FORADD TYPE_INCOMP_TABLE,
0497 1 TYPE_GRAPH_EDGE (D, G),0);
0498 1
0499 1
0500 1 Define the macro which declares a Type Hierarchy Table.
0501 1
0502 1 MACRO
0503 1 TYPE_INCOMP TABLE (TBLNAME)
0504 1 = BIND TBLNAME = PLIT WORD(%REMAINING) : VECTOR [,WORD] %;
0505 1
```

```
378 0506 1
379 0507 1
380 0508 1
381 0509 1
382 0510 1
383 0511 1
384 0512 1
385 0513 1
386 0514 1
387 0515 1
388 0516 1
389 0517 1
390 0518 1
391 0519 1
392 0520 1
393 0521 1
394 0522 1
395 0523 1
396 0524 1
397 0525 1
398 0526 1
399 0527 1
400 0528 1
401 0529 1
402 M 0530 1
403 0531 1
404 0532 1
405 0533 1
406 M 0534 1
407 M 0535 1
408 M 0536 1
409 M 0537 1
410 M 0538 1
411 M 0539 1
412 M 0540 1
413 M 0541 1
414 M 0542 1
415 M 0543 1
416 M 0544 1
417 M 0545 1
418 0546 1

OPERATOR INFORMATION TABLE

The Operator Information Table for a language is a blockvector indexed by
operator code. For each operator, it gives the address of the corresponding
Operator Routine Table, Type Hierarchy Table, Type Incompatibility Table,
and a flag. All addresses are relative to TABLEBASE. An Operator
Information Table is declared as follows:

    OPERATOR INFO TABLE(TBLNAME,
        OPERATOR_INFO_ENTRY(OPCODE, ROUTTBL, HIERTBL, INCOMPTBL,
            FETCH_FLAG),
        OPERATOR_INFO_ENTRY(OPCODE, ROUTTBL, HIERTBL, INCOMPTBL,
            FETCH_FLAG));

Here TBLNAME is the name of the whole Operator Information Table. For each
operator accepted by the language, OPCODE is the operator code name for the
operator (this is automatically prefixed by 'TOKENSK_' by the macro).

Define the macros which declare Operator Information Tables.

MACRO
    OPERATOR INFO TABLE(TBLNAME) =
        OWN TBLNAME : OPINFOTABLE PSECT(DBG$PLIT) PRESET(%REMAINING) %;

MACRO
    OPERATOR INFO ENTRY (OPERATOR, ROUTTBL, HIERTBL, INCOMPTBL, FALSE) =
        [ XNAME ('TOKENSK_', OPERATOR), OPINFOSL_ROUTTBL] =
            ROUTTBL - TABLEBASE,
        [ XNAME ('TOKENSK_', OPERATOR), OPINFOSL_HIERTBL] =
            HIERTBL - TABLEBASE,
        [ XNAME ('TOKENSK_', OPERATOR), OPINFOSL_INCOMPTBL] =
            INCOMPTBL - TABLEBASE,
        %IF %LENGTH LEQ 4
        %THEN
            [ XNAME ('TOKENSK_', OPERATOR), OPINFO$V_FETCH] = TRUE
        %ELSE
            [ XNAME ('TOKENSK_', OPERATOR), OPINFO$V_FETCH] = FALSE
        %FI %;
```

```
420 0547 1 TYPE MAPPING TABLE
421 0548 1
422 0549 1 The Type Mapping Table specifies the mapping of data types to be done
423 0550 1 prior to any expression evaluation. For example, in FORTRAN data type
424 0551 1 WU is treated the same as type W in expressions, so the mapping from
425 0552 1 WU to W is done here.
426 0553 1
427 0554 1 A Type Mapping Table is represented as a sequence of pairs of data types.
428 0555 1
429 0556 1 An example of a Type Mapping Table definition is:
430 0557 1
431 0558 1 TYPE_MAPPING_TABLE ( FORTRAN TYPE_MAPPING_TABLE,
432 0559 1     TYPE_GRAPH_EDGE (BU, BT),
433 0560 1     TYPE_GRAPH_EDGE (WU, W),
434 0561 1     TYPE_GRAPH_EDGE (LU, L),
435 0562 1     0);
436 0563 1
437 0564 1
438 0565 1 Define the macro which declares a Type Mapping Table.
439 0566 1
440 0567 1 MACRO
441 0568 1     TYPE_MAPPING_TABLE (TBLNAME)
442 0569 1         = BIND TBLNAME = PLIT WORD (%REMAINING) : VECTOR [,WORD] %;
443 0570 1
```

```
445 0571 1 TYPE CONVERSION INFORMATION TABLE
446 0572 1
447 0573 1
448 0574 1
449 0575 1 The Type Conversion Table for a language is a vector of longwords. For
450 0576 1 each longword, it gives the address of the corresponding Type Mapping Table,
451 0577 1 Type Conversion Table needed for the language. All addresses are relative
452 0578 1 to TABLEBASE. An Operator Information Table is declared as follows:
453 0579 1
454 0580 1     CONVERSION INFO TABLE(TBLNAME,
455 0581 1         CONVERSION_INFO_ENTRY(MAPTBL, CONVTL, CVT_ROUNDING_FLAG));
456 0582 1
457 0583 1 Here TBLNAME is the name of the whole Type Conversion Information Table.
458 0584 1
459 0585 1 Define the macros which declare Type Conversion Information Tables.
460 0586 1
461 0587 1 MACRO
462 M 0588 1     CONVERSION INFO TABLE(TBLNAME) =
463 0589 1         OWN TBLNAME: CVTINFOSTABLE PSECT(DBG$PLIT) PRESET(%REMAINING) %;
464 0590 1
465 0591 1 MACRO
466 M 0592 1     CONVERSION INFO ENTRY (MAPTBL, CVTTBL, TRUE) =
467 M 0593 1         [CVTINFO$MAPTBL] = MAPTBL - TABLEBASE,
468 M 0594 1         [CVTINFO$CVTTBL] = CVTTBL - TABLEBASE,
469 M 0595 1         %IF %LENGTH LEQ 2
470 M 0596 1         %THEN
471 M 0597 1             [CVTINFO$V_ROUND] = FALSE
472 M 0598 1         %ELSE
473 M 0599 1             [CVTINFO$V_ROUND] = TRUE
474 0600 1         %FI %;
```

```
476 0601 1 | LANGUAGE SPECIFIC TYPE CONVERSION TABLE
477 0602 1 |
478 0603 1 | The Language Specific Type Conversion Table specifies which type conversions
479 0604 1 | are to be done in a language specific manner. For example, although PL/I
480 0605 1 | bit-strings have a dtype of V or VU, they are handled very differently from
481 0606 1 | most other data having those data types (as in BLISS or Pascal).
482 0607 1 |
483 0608 1 | A language specific conversion table is specified by giving a case index,
484 0609 1 | and a higher and lower type.
485 0610 1 |
486 0611 1 | An example of a language specific conversion table entry is:
487 0612 1 |     LANG_CVT_TABLE (PLI_CVT_TABLE
488 0613 1 |         [LANG_CVT_INDEX (CVT$PLI_TF_L, L, TF),
489 0614 1 |         0]);
490 0615 1 |
491 0616 1 |
492 0617 1 |
493 0618 1 | Define the macro which declares the entries for Language Specific Type
494 0619 1 | Conversion Table.
495 0620 1 |
496 0621 1 | MACRO
497 0622 1 |     LANG_CVT_ENTRY (INDEX, LOWER_TYPE, HIGHER_TYPE) =
498 0623 1 |         T(((XNAME ('CVT$K_', INDEX))^8)
499 0624 1 |         OR
500 0625 1 |         (XIF XDECLARED (XNAME ('RST$K_TYPE_', HIGHER_TYPE))
501 0626 1 |         XTHEN
502 0627 1 |             DBG$K_MAXIMUM_DTYPE + XNAME ('RST$K_TYPE_', HIGHER_TYPE)
503 0628 1 |         XELSE
504 0629 1 |             XNAME ('DSC$K_DTYPE_', HIGHER_TYPE)
505 0630 1 |         XFI))^8)
506 0631 1 |         OR
507 0632 1 |         (XIF XDECLARED (XNAME ('RST$K_TYPE_', LOWER_TYPE))
508 0633 1 |         XTHEN
509 0634 1 |             DBG$K_MAXIMUM_DTYPE + XNAME ('RST$K_TYPE_', LOWER_TYPE)
510 0635 1 |         XELSE
511 0636 1 |             XNAME ('DSC$K_DTYPE_', LOWER_TYPE)
512 0637 1 |         XFI) X;
513 0638 1 |
514 0639 1 |
515 0640 1 | Define the macro which declares a Language Specific Type Conversion Table.
516 0641 1 |
517 0642 1 | MACRO
518 0643 1 |     LANG_CVT_TABLE (TBLNAME)
519 0644 1 |     = BIRD TBLNAME = PLIT LONG (XREMAINING): VECTOR[, LONG] X;
520 0645 1 |
```

```
0646 1 | ADA OPERATOR INFORMATION TABLES
0647 1 |
0648 1 | This section contains the Operator Routine and Type tables needed to
0649 1 | evaluate expressions in the ADA language.
0650 1 |
0651 1 |
0652 1 | Define the Type Conversion Information Table for ADA.
0653 1 | For now, there are no exceptions to the standard DBG$CVT_DX_DX conversion
0654 1 | rules so we do not specify any tables here.
0655 1 |
P 0656 1 | CONVERSION_INFO_TABLE (ADA_CVTINFO_TABLE,
0657 1 | CONVERSION_INFO_ENTRY (TABLEBASE, TABLEBASE));
0658 1 |
0659 1 |
0660 1 | Define the Type Hierarchy Table for ADA. This table specifies the implicit
0661 1 | type conversions to be done on arithmetic operations.
0662 1 |
0663 1 | For debugging purposes, we have decided to adopting looser rules than
0664 1 | the compiler. For example, to add an integer X to a float Y in ADA,
0665 1 | you have to say something like "FLOAT(X) + Y". But in the debugger, we
0666 1 | will do the implicit conversion and allow just "X + Y".
0667 1 |
0668 1 | We thus allow integer to be converted to either FIXED or FLOAT.
0669 1 | We allow the conversion of FIXED to H float (but not to F, D, or G
0670 1 | since that could lose precision).
0671 1 |
0672 1 |      BU, WU, LU          G
0673 1 |      B, W ----> L --> F --> D --> H
0674 1 |      |               / \
0675 1 |      |               - FIXED -
0676 1 |
P 0677 1 | TYPE_HIERARCHY_TABLE (ADA_HIER_TABLE,
0678 1 | TYPE_GRAPH_EDGE (B, L),
0679 1 | TYPE_GRAPH_EDGE (BU, L),
0680 1 | TYPE_GRAPH_EDGE (W, L),
0681 1 | TYPE_GRAPH_EDGE (WU, L),
0682 1 | TYPE_GRAPH_EDGE (LU, L),
0683 1 | TYPE_GRAPH_EDGE (L, F),
0684 1 | TYPE_GRAPH_EDGE (L, FIXED),
0685 1 | TYPE_GRAPH_EDGE (F, D),
0686 1 | TYPE_GRAPH_EDGE (F, G),
0687 1 | TYPE_GRAPH_EDGE (FIXED, H),
0688 1 | TYPE_GRAPH_EDGE (D, H),
0689 1 | TYPE_GRAPH_EDGE (G, H),
0690 1 | 0);
0691 1 |
0692 1 |
0693 1 |
0694 1 | Define the Type Hierarchy Table for ADA deposit.
0695 1 |
P 0696 1 | TYPE_HIERARCHY_TABLE (ADA_DEPOSIT_TABLE,
0697 1 |
0698 1 | | Allow any numeric type to any other numeric type.
0699 1 |
P 0700 1 | TYPE_GRAPH_EDGE (B, BU),
P 0701 1 | TYPE_GRAPH_EDGE (BU, W),
P 0702 1 | TYPE_GRAPH_EDGE (W, WU).
```

```
579 P 0703 1 TYPE_GRAPH_EDGE (WU, L),
580 P 0704 1 TYPE_GRAPH_EDGE (L, LU),
581 P 0705 1 TYPE_GRAPH_EDGE (LU, FIXED),
582 P 0706 1 TYPE_GRAPH_EDGE (FIXED, F),
583 P 0707 1 TYPE_GRAPH_EDGE (F, D),
584 P 0708 1 TYPE_GRAPH_EDGE (D, G),
585 P 0709 1 TYPE_GRAPH_EDGE (G, H),
586 P 0710 1 TYPE_GRAPH_EDGE (H, B),
587 P 0711 1
588 P 0712 1 ! Allow number -> SUBRNG. The constraint check should be done
589 P 0713 1 ! by the DBG$PERFORM TYPEID_CHECK routine. Also allow ENUM->SUBRNG
590 P 0714 1 ! and character -> SUBRNG.
591 P 0715 1
592 P 0716 1 TYPE_GRAPH_EDGE (ENUM, SUBRNG),
593 P 0717 1 TYPE_GRAPH_EDGE (T, SUBRNG),
594 P 0718 1 TYPE_GRAPH_EDGE (L, SUBRNG),
595 P 0719 1 TYPE_GRAPH_EDGE (FIXED, SUBRNG),
596 P 0720 1 TYPE_GRAPH_EDGE (F, SUBRNG),
597 P 0721 1 TYPE_GRAPH_EDGE (D, SUBRNG),
598 P 0722 1 TYPE_GRAPH_EDGE (G, SUBRNG),
599 P 0723 1 TYPE_GRAPH_EDGE (H, SUBRNG),
600 P 0724 1 0);
601 P 0725 1
602 P 0726 1
603 P 0727 1 ! Define the Operator Routine Table for ADA unary plus.
604 P 0728 1
605 P 0729 1 OPERATOR_ROUTINE_TABLE (ADA_UNARY_PLUS_TABLE,
606 P 0730 1
607 P 0731 1 ! The following are not language dependent types. This is needed for DEBUG
608 P 0732 1 ! types. For example, DEP/QUAD L= +1.
609 P 0733 1
610 P 0734 1 OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
611 P 0735 1 OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
612 P 0736 1 OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
613 P 0737 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
614 P 0738 1 OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
615 P 0739 1
616 P 0740 1 OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
617 P 0741 1 OPERATOR_ROUTINE (FIXED, FIXED, FIXED, UNARY_PLUS_FIXED),
618 P 0742 1 OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
619 P 0743 1 OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
620 P 0744 1 OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
621 P 0745 1 OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H);
622 P 0746 1
623 P 0747 1
624 P 0748 1 ! Define the Operator Routine Table for ADA unary minus.
625 P 0749 1
626 P 0750 1 OPERATOR_ROUTINE_TABLE (ADA_UNARY_MINUS_TABLE,
627 P 0751 1
628 P 0752 1 ! The following are not language dependent types. This is needed for DEBUG
629 P 0753 1 ! types. For example, DEP/QUAD L= -1.
630 P 0754 1
631 P 0755 1 OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
632 P 0756 1 OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
633 P 0757 1 OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
634 P 0758 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
635 P 0759 1 OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
```

```
636 P 0760 1
637 P 0761 1 OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
638 P 0762 1 OPERATOR_ROUTINE (FIXED, FIXED, FIXED, UNARY_MINUS_FIXED),
639 P 0763 1 OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
640 P 0764 1 OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
641 P 0765 1 OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
642 P 0766 1 OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H);
643 P 0767 1
644 P 0768 1
645 P 0769 1 ! Define the Operator Routine Table for ADA absolute value.
646 P 0770 1
647 P 0771 1 OPERATOR_ROUTINE_TABLE (ADA_ABSOLUTE_TABLE,
648 P 0772 1 OPERATOR_ROUTINE (L, L, L, ABS_L),
649 P 0773 1 OPERATOR_ROUTINE (FIXED, FIXED, FIXED, ABS_FIXED),
650 P 0774 1 OPERATOR_ROUTINE (F, F, F, ABS_F),
651 P 0775 1 OPERATOR_ROUTINE (D, D, D, ABS_D),
652 P 0776 1 OPERATOR_ROUTINE (G, G, G, ABS_G),
653 P 0777 1 OPERATOR_ROUTINE (H, H, H, ABS_H);
654 P 0778 1
655 P 0779 1
656 P 0780 1 ! Define the Operator Routine Table for ADA addition.
657 P 0781 1
658 P 0782 1 OPERATOR_ROUTINE_TABLE (ADA_ADD_TABLE,
659 P 0783 1 OPERATOR_ROUTINE (L, L, L, ADD_L_L),
660 P 0784 1 OPERATOR_ROUTINE (FIXED, FIXED, FIXED, ADD_FIXED_FIXED),
661 P 0785 1 OPERATOR_ROUTINE (F, F, F, ADD_F_F),
662 P 0786 1 OPERATOR_ROUTINE (D, D, D, ADD_D_D),
663 P 0787 1 OPERATOR_ROUTINE (G, G, G, ADD_G_G),
664 P 0788 1 OPERATOR_ROUTINE (H, H, H, ADD_H_H);
665 P 0789 1
666 P 0790 1
667 P 0791 1 ! Define the Operator Routine Table for ADA subtraction.
668 P 0792 1
669 P 0793 1 OPERATOR_ROUTINE_TABLE (ADA_SUBTRACT_TABLE,
670 P 0794 1 OPERATOR_ROUTINE (L, L, L, SUB_L_L),
671 P 0795 1 OPERATOR_ROUTINE (FIXED, FIXED, FIXED, SUB_FIXED_FIXED),
672 P 0796 1 OPERATOR_ROUTINE (F, F, F, SUB_F_F),
673 P 0797 1 OPERATOR_ROUTINE (D, D, D, SUB_D_D),
674 P 0798 1 OPERATOR_ROUTINE (G, G, G, SUB_G_G),
675 P 0799 1 OPERATOR_ROUTINE (H, H, H, SUB_H_H);
676 P 0800 1
677 P 0801 1
678 P 0802 1 ! Define the Operator Routine Table for ADA multiplication.
679 P 0803 1
680 P 0804 1 OPERATOR_ROUTINE_TABLE (ADA_MULTIPLY_TABLE,
681 P 0805 1 OPERATOR_ROUTINE (L, L, L, MUL_L_L),
682 P 0806 1 OPERATOR_ROUTINE (FIXED, FIXED, FIXED, MUL_FIXED_FIXED),
683 P 0807 1 OPERATOR_ROUTINE (F, F, F, MUL_F_F),
684 P 0808 1 OPERATOR_ROUTINE (D, D, D, MUL_D_D),
685 P 0809 1 OPERATOR_ROUTINE (G, G, G, MUL_G_G),
686 P 0810 1 OPERATOR_ROUTINE (H, H, H, MUL_H_H);
687 P 0811 1
688 P 0812 1
689 P 0813 1 ! Define the Operator Routine Table for ADA division.
690 P 0814 1
691 P 0815 1 OPERATOR_ROUTINE_TABLE (ADA_DIVIDE_TABLE,
692 P 0816 1 OPERATOR_ROUTINE (L, L, L, DIV_L_L),
```

```
693 P 0817 1 OPERATOR_ROUTINE (FIXED, FIXED, FIXED, DIV_FIXED_FIXED),
694 P 0818 1 OPERATOR_ROUTINE (F, F, F, DIV_F_F),
695 P 0819 1 OPERATOR_ROUTINE (D, D, D, DIV_D_D),
696 P 0820 1 OPERATOR_ROUTINE (G, G, G, DIV_G_G),
697 0821 1 OPERATOR_ROUTINE (H, H, H, DIV_H_H);
698 0822 1
699 0823 1
700 0824 1 ! Define the Operator Routine Table for ADA Modulus.
701 0825 1
702 P 0826 1 OPERATOR_ROUTINE_TABLE (ADA_MODULUS_TABLE,
703 0827 1 OPERATOR_ROUTINE (L, L, L, MOD_L_L));
704 0828 1
705 0829 1
706 0830 1 ! Define the Operator Routine Table for ADA Remainder.
707 0831 1
708 P 0832 1 OPERATOR_ROUTINE_TABLE (ADA_REMAINDER_TABLE,
709 0833 1 OPERATOR_ROUTINE (L, L, L, REM_L_L));
710 0834 1
711 0835 1
712 0836 1 ! Define the Operator Routine Table for ADA Exponentiation.
713 0837 1
714 P 0838 1 OPERATOR_ROUTINE_TABLE (ADA_POWER_OF_TABLE,
715 P 0839 1 OPERATOR_ROUTINE (L, L, L, POWER_L_L),
716 P 0840 1 OPERATOR_ROUTINE (F, F, F, POWER_F_F),
717 P 0841 1 OPERATOR_ROUTINE (D, D, D, POWER_D_D),
718 P 0842 1 OPERATOR_ROUTINE (G, G, G, POWER_G_G),
719 0843 1 OPERATOR_ROUTINE (H, H, H, POWER_H_H);
720 0844 1
721 0845 1
722 0846 1 ! Define the Operator Routine Table for ADA Logical Not.
723 0847 1
724 P 0848 1 OPERATOR_ROUTINE_TABLE (ADA_NOT_TABLE,
725 0849 1 OPERATOR_ROUTINE (TF, TF, TF, NOT_L));
726 0850 1
727 0851 1
728 0852 1 ! Define the Operator Routine Table for ADA Logical And.
729 0853 1
730 P 0854 1 OPERATOR_ROUTINE_TABLE (ADA_AND_TABLE,
731 0855 1 OPERATOR_ROUTINE (TF, TF, TF, AND_L_L));
732 0856 1
733 0857 1
734 0858 1 ! Define the Operator Routine Table for ADA Logical Or.
735 0859 1
736 P 0860 1 OPERATOR_ROUTINE_TABLE (ADA_OR_TABLE,
737 0861 1 OPERATOR_ROUTINE (TF, TF, TF, OR_L_L));
738 0862 1
739 0863 1
740 0864 1 ! Define the Operator Routine Table for ADA Logical Exclusive Or.
741 0865 1
742 P 0866 1 OPERATOR_ROUTINE_TABLE (ADA_XOR_TABLE,
743 0867 1 OPERATOR_ROUTINE (TF, TF, TF, XOR_L_L));
744 0868 1
745 0869 1
746 0870 1 ! Define the Operator Routine Table for ADA Equal.
747 0871 1
748 P 0872 1 OPERATOR_ROUTINE_TABLE (ADA_EQUAL_TABLE,
749 P 0873 1 OPERATOR_ROUTINE (ENUM, ENUM, TF, EQL_L_L, ENUM_ENUM),
```

```
750 P 0874 1 OPERATOR_ROUTINE (L, L, TF, EQL_L_L),
751 P 0875 1 OPERATOR_ROUTINE (FIXED, FIXED, TF, EQL_FIXED_FIXED),
752 P 0876 1 OPERATOR_ROUTINE (F, F, TF, EQL_F_F),
753 P 0877 1 OPERATOR_ROUTINE (D, D, TF, EQL_D_D),
754 P 0878 1 OPERATOR_ROUTINE (G, G, TF, EQL_G_G),
755 P 0879 1 OPERATOR_ROUTINE (H, H, TF, EQL_H_H),
756 P 0880 1 OPERATOR_ROUTINE (TF, TF, TF, EQL_TF_TF));
757 0881 1
758 0882 1
759 0883 1 !! Define the Operator Routine Table for ADA Not Equal.
760 0884 1
761 P 0885 1 OPERATOR_ROUTINE_TABLE (ADA_NOT_EQUAL_TABLE,
762 P 0886 1 OPERATOR_ROUTINE (ENUM, ENUM, TF, NEQ_L_L, ENUM_ENUM),
763 P 0887 1 OPERATOR_ROUTINE (L, L, TF, NEQ_L_L),
764 P 0888 1 OPERATOR_ROUTINE (FIXED, FIXED, TF, NEQ_FIXED_FIXED),
765 P 0889 1 OPERATOR_ROUTINE (F, F, TF, NEQ_F_F),
766 P 0890 1 OPERATOR_ROUTINE (D, D, TF, NEQ_D_D),
767 P 0891 1 OPERATOR_ROUTINE (G, G, TF, NEQ_G_G),
768 P 0892 1 OPERATOR_ROUTINE (H, H, TF, NEQ_H_H),
769 0893 1 OPERATOR_ROUTINE (TF, TF, TF, NEQ_TF_TF));
770 0894 1
771 0895 1
772 0896 1 !! Define the Operator Routine Table for ADA Less Than.
773 0897 1
774 P 0898 1 OPERATOR_ROUTINE_TABLE (ADA_LSS_THAN_TABLE,
775 P 0899 1 OPERATOR_ROUTINE (ENUM, ENUM, TF, LSS_L_L, ENUM_ENUM),
776 P 0900 1 OPERATOR_ROUTINE (L, L, TF, LSS_L_L),
777 P 0901 1 OPERATOR_ROUTINE (FIXED, FIXED, TF, LSS_FIXED_FIXED),
778 P 0902 1 OPERATOR_ROUTINE (F, F, TF, LSS_F_F),
779 P 0903 1 OPERATOR_ROUTINE (D, D, TF, LSS_D_D),
780 P 0904 1 OPERATOR_ROUTINE (G, G, TF, LSS_G_G),
781 P 0905 1 OPERATOR_ROUTINE (H, H, TF, LSS_H_H),
782 0906 1 OPERATOR_ROUTINE (TF, TF, TF, LSS_TF_TF));
783 0907 1
784 0908 1
785 0909 1 !! Define the Operator Routine Table for ADA Greater Than.
786 0910 1
787 P 0911 1 OPERATOR_ROUTINE_TABLE (ADA_GTR_THAN_TABLE,
788 P 0912 1 OPERATOR_ROUTINE (ENUM, ENUM, TF, GTR_L_L, ENUM_ENUM),
789 P 0913 1 OPERATOR_ROUTINE (L, L, TF, GTR_L_L),
790 P 0914 1 OPERATOR_ROUTINE (FIXED, FIXED, TF, GTR_FIXED_FIXED),
791 P 0915 1 OPERATOR_ROUTINE (F, F, TF, GTR_F_F),
792 P 0916 1 OPERATOR_ROUTINE (D, D, TF, GTR_D_D),
793 P 0917 1 OPERATOR_ROUTINE (G, G, TF, GTR_G_G),
794 P 0918 1 OPERATOR_ROUTINE (H, H, TF, GTR_H_H),
795 0919 1 OPERATOR_ROUTINE (TF, TF, TF, GTR_TF_TF));
796 0920 1
797 0921 1
798 0922 1 !! Define the Operator Routine Table for ADA Less Than or Equal to.
799 0923 1
800 P 0924 1 OPERATOR_ROUTINE_TABLE (ADA_LSS_EQUAL_TABLE,
801 P 0925 1 OPERATOR_ROUTINE (ENUM, ENUM, TF, LEQ_L_L, ENUM_ENUM),
802 P 0926 1 OPERATOR_ROUTINE (L, L, TF, LEQ_L_L),
803 P 0927 1 OPERATOR_ROUTINE (FIXED, FIXED, TF, LEQ_FIXED_FIXED),
804 P 0928 1 OPERATOR_ROUTINE (F, F, TF, LEQ_F_F),
805 P 0929 1 OPERATOR_ROUTINE (D, D, TF, LEQ_D_D),
806 P 0930 1 OPERATOR_ROUTINE (G, G, TF, LEQ_G_G),
```

```
807 P 0931 1 OPERATOR_ROUTINE (H, H, TF, LEQ_H_H);
808 P 0932 1 OPERATOR_ROUTINE (TF, TF, TF, LEQ_TF_TF));
809 P 0933 1
810 P 0934 1
811 P 0935 1 ! Define the Operator Routine Table for ADA Greater Than or Equal to.
812 P 0936 1
813 P 0937 1 OPERATOR_ROUTINE_TABLE (ADA_GTR_EQUAL_TABLE,
814 P 0938 1 OPERATOR_ROUTINE (ENUM, ENUM, TF, GEQ_L_L, ENUM_ENUM),
815 P 0939 1 OPERATOR_ROUTINE (L, L, TF, GEQ_L_L),
816 P 0940 1 OPERATOR_ROUTINE (FIXED, FIXED, TF, GEQ_FIXED_FIXED),
817 P 0941 1 OPERATOR_ROUTINE (F, F, TF, GEQ_F_F),
818 P 0942 1 OPERATOR_ROUTINE (D, D, TF, GEQ_D_D),
819 P 0943 1 OPERATOR_ROUTINE (G, G, TF, GEQ_G_G),
820 P 0944 1 OPERATOR_ROUTINE (H, H, TF, GEQ_H_H),
821 P 0945 1 OPERATOR_ROUTINE (TF, TF, TF, GEQ_TF_TF));
822 P 0946 1
823 P 0947 1
824 P 0948 1 ! Define the Operator Routine Table for ADA Concatenate.
825 P 0949 1
826 P 0950 1 OPERATOR_ROUTINE_TABLE (ADA_CONCATENATE_TABLE,
827 P 0951 1 OPERATOR_ROUTINE (T, T, T, CONCAT_T_T));
828 P 0952 1
829 P 0953 1
830 P 0954 1 ! Define the Operator Information Table for ADA.
831 P 0955 1
832 P 0956 1 OPERATOR_INFO_TABLE (ADA_OPINFO_TABLE,
833 P 0957 1
834 P 0958 1 ! Unary arithmetic.
835 P 0959 1
836 P 0960 1 OPERATOR_INFO_ENTRY
837 P 0961 1 (UNARY_PLUS, ADA_UNARY_PLUS_TABLE, ADA_HIER_TABLE, TABLEBASE),
838 P 0962 1 OPERATOR_INFO_ENTRY
839 P 0963 1 (UNARY_MINUS, ADA_UNARY_MINUS_TABLE, ADA_HIER_TABLE, TABLEBASE),
840 P 0964 1 OPERATOR_INFO_ENTRY
841 P 0965 1 (ABSOLUTE, ADA_ABSOLUTE_TABLE, ADA_HIER_TABLE, TABLEBASE),
842 P 0966 1
843 P 0967 1 ! Binary arithmetic.
844 P 0968 1
845 P 0969 1 OPERATOR_INFO_ENTRY
846 P 0970 1 (ADD, ADA_ADD_TABLE, ADA_HIER_TABLE, TABLEBASE),
847 P 0971 1 OPERATOR_INFO_ENTRY
848 P 0972 1 (SUBTRACT, ADA_SUBTRACT_TABLE, ADA_HIER_TABLE, TABLEBASE),
849 P 0973 1 OPERATOR_INFO_ENTRY
850 P 0974 1 (MULTIPLY, ADA_MULTIPLY_TABLE, ADA_HIER_TABLE, TABLEBASE),
851 P 0975 1 OPERATOR_INFO_ENTRY
852 P 0976 1 (DIVIDE, ADA_DIVIDE_TABLE, ADA_HIER_TABLE, TABLEBASE),
853 P 0977 1 OPERATOR_INFO_ENTRY
854 P 0978 1 (MODULUS, ADA_MODULUS_TABLE, ADA_HIER_TABLE, TABLEBASE),
855 P 0979 1 OPERATOR_INFO_ENTRY
856 P 0980 1 (REMAINDER, ADA_REMAINDER_TABLE, ADA_HIER_TABLE, TABLEBASE),
857 P 0981 1 OPERATOR_INFO_ENTRY
858 P 0982 1 (POWER_OF, ADA_POWER_OF_TABLE, ADA_HIER_TABLE, TABLEBASE),
859 P 0983 1
860 P 0984 1 ! Logical operations.
861 P 0985 1
862 P 0986 1 OPERATOR_INFO_ENTRY
863 P 0987 1 (NOT, ADA_NOT_TABLE, ADA_HIER_TABLE, TABLEBASE),
```

```
... 864 P 0988 1 OPERATOR_INFO_ENTRY
      865 P 0989 1 (AND, ADA_AND_TABLE, ADA_HIER_TABLE, TABLEBASE),
      866 P 0990 1 OPERATOR_INFO_ENTRY
      867 P 0991 1 (OR, ADA_OR_TABLE, ADA_HIER_TABLE, TABLEBASE),
      868 P 0992 1 OPERATOR_INFO_ENTRY
      869 P 0993 1 (XOR, ADA_XOR_TABLE, ADA_HIER_TABLE, TABLEBASE),
      870 P 0994 1
      871 P 0995 1 ! Relations.
      872 P 0996 1
      873 P 0997 1 OPERATOR_INFO_ENTRY
      874 P 0998 1 (EQUAL, ADA_EQUAL_TABLE, ADA_HIER_TABLE, TABLEBASE),
      875 P 0999 1 OPERATOR_INFO_ENTRY
      876 P 1000 1 (NOT_EQUAL, ADA_NOT_EQUAL_TABLE, ADA_HIER_TABLE, TABLEBASE),
      877 P 1001 1 OPERATOR_INFO_ENTRY
      878 P 1002 1 (LSS_THAN, ADA_LSS_THAN_TABLE, ADA_HIER_TABLE, TABLEBASE),
      879 P 1003 1 OPERATOR_INFO_ENTRY
      880 P 1004 1 (GTR_THAN, ADA_GTR_THAN_TABLE, ADA_HIER_TABLE, TABLEBASE),
      881 P 1005 1 OPERATOR_INFO_ENTRY
      882 P 1006 1 (LSS_EQUAL, ADA_LSS_EQUAL_TABLE, ADA_HIER_TABLE, TABLEBASE),
      883 P 1007 1 OPERATOR_INFO_ENTRY
      884 P 1008 1 (GTR_EQUAL, ADA_GTR_EQUAL_TABLE, ADA_HIER_TABLE, TABLEBASE),
      885 P 1009 1
      886 P 1010 1 ! String operations.
      887 P 1011 1
      888 P 1012 1 OPERATOR_INFO_ENTRY
      889 P 1013 1 (CONCATENATE, ADA_CONCATENATE_TABLE, ADA_HIER_TABLE, TABLEBASE),
      890 P 1014 1
      891 P 1015 1 ! Convert, Deposit, and Identity.
      892 P 1016 1
      893 P 1017 1 OPERATOR_INFO_ENTRY
      894 P 1018 1 (CONVERT, TABLEBASE, ADA_DEPOSIT_TABLE, TABLEBASE),
      895 P 1019 1 OPERATOR_INFO_ENTRY
      896 P 1020 1 (DEPOSIT, TABLEBASE, ADA_DEPOSIT_TABLE, TABLEBASE),
      897 P 1021 1 OPERATOR_INFO_ENTRY
      898 1022 1 (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE));
      899 1023 1
```

## BASIC OPERATOR INFORMATION TABLES

This section contains the Operator Routine and Type tables needed to evaluate expressions in the BASIC language.

The following summarizes the information in the BASIC manual about data types, type conversions, and operators. There is further documentation within the tables below, describing exactly how we translate this into the DEBUG tables.

## BASIC Data Types:

- . Integer (signed only)
  - byte (8-bit), word (16-bit), long (32-bit)
- . Real
  - single (f-float), double (d-float), gfloat, hfloat
- . Packed Decimal
  - 0-16 bytes; specifies number of digits and location of decimal point
- . String
  - one character per byte
- . RFA
  - 6 bytes; specifies record file address - block number and offset

## BASIC Constants:

- . Any of BASIC's data types

## BASIC Aggregates:

- . Array
- . Record

## Expressions:

- . Numeric
  - Floating-point or integer operands separated by arithmetic operators (+, -, \*, /, ^, \*\*) and optionally grouped by parentheses.
  - Result: numeric (see type conversion).
- . String
  - Strings separated by "+" (concatenation) or by combinations of string functions.
  - Result: string.
- . Relational
  - Operands may be either numeric or string (not mixed)
  - Operators: =, <, >, <= (or =<), >= (or =>), <> (or ><), ==  
(note: "==" is different for string and numeric)
  - Result: true (-1) or false (0)
- . Logical
  - Operands: integer only.
  - Operators: NOT, AND, OR, XOR, EQV, IMP
  - Result: true (-1) or false (0)
- . Assignment, conditional

## Type Conversion:

- . Arithmetic
  - Note that, with one exception, the resulting data type is the same as that of the operand with the higher data type. The exception is when the operands are DOUBLE and GFLOAT: BASIC promotes both values to HFLOAT, and the result is HFLOAT. This preserves both precision

and magnitude.

	!	BYTE	WORD	LONG	SINGLE	DOUBLE	GFLOAT	HFLOAT
-----	-----	-----	-----	-----	-----	-----	-----	-----
BYTE	!	BYTE	WORD	LONG	SINGLE	DOUBLE	GFLOAT	HFLOAT
-----	-----	-----	-----	-----	-----	-----	-----	-----
WORD	!	WORD	WORD	LONG	SINGLE	DOUBLE	GFLOAT	HFLOAT
-----	-----	-----	-----	-----	-----	-----	-----	-----
LONG	!	LONG	LONG	LONG	SINGLE	DOUBLE	GFLOAT	HFLOAT
-----	-----	-----	-----	-----	-----	-----	-----	-----
SINGLE	!	SINGLE	SINGLE	SINGLE	SINGLE	DOUBLE	GFLOAT	HFLOAT
-----	-----	-----	-----	-----	-----	-----	-----	-----
DOUBLE	!	DOUBLE	DOUBLE	DOUBLE	DOUBLE	DOUBLE	HFLOAT	HFLOAT
-----	-----	-----	-----	-----	-----	-----	-----	-----
GFLOAT	!	GFLOAT	GFLOAT	GFLOAT	GFLOAT	HFLOAT	GFLOAT	HFLOAT
-----	-----	-----	-----	-----	-----	-----	-----	-----
HFLOAT	!	HFLOAT	HFLOAT	HFLOAT	HFLOAT	HFLOAT	HFLOAT	HFLOAT

#### . Packed Decimal Conversion

1. If both operands are Decimal with the same digit and scale values, no conversion is performed. If the operands have different digit and scale values, BASIC always used the larger number of specified digits for the result. The debugger uses a large enough digit and scale factor to avoid overflows if possible.

2. If one operand is Decimal and one is integer, the following integer --> decimal conversions occur (in BASIC):

BYTE --> DECIMAL(3,0)  
WORD --> DECIMAL(5,0)  
LONG --> DECIMAL(10,0)

The debugger converts them all to DECIMAL(31,0).

3. If one operand is Decimal(d,s) and one is floating-point, the following decimal --> floating-point conversions occur:

if range of d is <=1 thru <=6 --> SINGLE  
if range of d is <=7 thru <=15 --> DOUBLE  
if range of d is <=16 --> GFLOAT  
if range of d is <=17 thru <=31 --> HFLOAT

-- depends on floating-point operand

The debugger employs this same scheme.

Define the Type Conversion Information Table for BASIC.

There is no CVT\_TABLE specifying exceptions to the DBG\$CVT\_DX\_DX rules.

CONVERSION INFO TABLE (BASIC CVTINFO\_TABLE,  
CONVERSION\_INFO\_ENTRY (TABLEBASE, TABLEBASE));

Define the Type Hierarchy Table for BASIC.

This table is described above when we talk about conversion rules.

Leaving out the G edges, the graph specified by this table is:

B -> W -> L -> F -> D -> H

```
1015 1138 1 | |
1016 1139 1 | |
1017 1140 1 | | The case of converting packed to float is handled by the routine
1018 1141 1 | | MAP_PACKED in DBGEVALOP.
1019 1142 1 | |
1020 1143 1 | |
1021 1144 1 | | Define a Type Hierarachy Table for BASIC.
1022 1145 1 | |
1023 1146 1 | | TYPE_HIERARCHY_TABLE (BASIC_HIER1_TABLE,
1024 1147 1 | | TYPE_GRAPH_EDGE (B, W),
1025 1148 1 | | TYPE_GRAPH_EDGE (W, L),
1026 1149 1 | | TYPE_GRAPH_EDGE (L, F),
1027 1150 1 | | TYPE_GRAPH_EDGE (L, P),
1028 1151 1 | | TYPE_GRAPH_EDGE (F, D),
1029 1152 1 | | TYPE_GRAPH_EDGE (F, G),
1030 1153 1 | | TYPE_GRAPH_EDGE (D, H),
1031 1154 1 | | TYPE_GRAPH_EDGE (G, H),
1032 1155 1 | | 0);
1033 1156 1 | |
1034 1157 1 | |
1035 1158 1 | | Define another Type Hierarachy Table for BASIC.
1036 1159 1 | | This is a subset of the HIER1 table. It is used for those operators that
1037 1160 1 | | only accept integer types. It would also be OK to use the HIER1 table,
1038 1161 1 | | but providing a smaller table speeds up the code.
1039 1162 1 | |
1040 1163 1 | | TYPE_HIERARCHY_TABLE (BASIC_HIER2_TABLE,
1041 1164 1 | | TYPE_GRAPH_EDGE (B, W),
1042 1165 1 | | TYPE_GRAPH_EDGE (W, L),
1043 1166 1 | | 0);
1044 1167 1 | |
1045 1168 1 | | Define the Type Hierarchy Table for BASIC deposit.
1046 1169 1 | | This is a circular table which includes all types except T.
1047 1170 1 | | This means that any of the numeric types are convertible to any
1048 1171 1 | | of the other numeric types on a DEPOSIT.
1049 1172 1 | |
1050 1173 1 | | TYPE_HIERARCHY_TABLE (BASIC_HIERD_TABLE,
1051 1174 1 | | TYPE_GRAPH_EDGE (B, W),
1052 1175 1 | | TYPE_GRAPH_EDGE (W, L),
1053 1176 1 | | TYPE_GRAPH_EDGE (L, P),
1054 1177 1 | | TYPE_GRAPH_EDGE (P, F),
1055 1178 1 | | TYPE_GRAPH_EDGE (F, D),
1056 1179 1 | | TYPE_GRAPH_EDGE (D, G),
1057 1180 1 | | TYPE_GRAPH_EDGE (G, H),
1058 1181 1 | | TYPE_GRAPH_EDGE (H, B),
1059 1182 1 | | 0);
1060 1183 1 | |
1061 1184 1 | |
1062 1185 1 | | ++
1063 1186 1 | | Most of the arithmetic routines operate on two arguments of the same type.
1064 1187 1 | | That type may be B, W, L, F, D, G, H, P, so we provide all of those case
1065 1188 1 | | indices.
1066 1189 1 | | --
1067 1190 1 | |
1068 1191 1 | | Define the Operator Routine Table for BASIC addition.
1069 1192 1 | | Note that addition of text strings is actually concatenation.
1070 1193 1 | |
1071 1194 1 | | OPERATOR_ROUTINE_TABLE (BASIC_ADD_TABLE,
```

```
1072 P 1195 1 OPERATOR_ROUTINE (B, B, B, ADD_B_B);
1073 P 1196 1 OPERATOR_ROUTINE (W, W, W, ADD_W_W);
1074 P 1197 1 OPERATOR_ROUTINE (L, L, L, ADD_L_L);
1075 P 1198 1 OPERATOR_ROUTINE (F, F, F, ADD_F_F);
1076 P 1199 1 OPERATOR_ROUTINE (D, D, D, ADD_D_D);
1077 P 1200 1 OPERATOR_ROUTINE (G, G, G, ADD_G_G);
1078 P 1201 1 OPERATOR_ROUTINE (H, H, H, ADD_H_H);
1079 P 1202 1 OPERATOR_ROUTINE (P, P, P, ADD_P_P);
1080 1203 1 OPERATOR_ROUTINE (T, T, T, CONCAT_T_T));
1081 1204 1
1082 1205 1
1083 1206 1 ! Define the Operator Routine Table for BASIC subtraction.
1084 1207 1 !
1085 P 1208 1 OPERATOR_ROUTINE_TABLE (BASIC_SUB_TABLE,
1086 P 1209 1 OPERATOR_ROUTINE (B, B, B, SUB_B_B);
1087 P 1210 1 OPERATOR_ROUTINE (W, W, W, SUB_W_W);
1088 P 1211 1 OPERATOR_ROUTINE (L, L, L, SUB_L_L);
1089 P 1212 1 OPERATOR_ROUTINE (F, F, F, SUB_F_F);
1090 P 1213 1 OPERATOR_ROUTINE (D, D, D, SUB_D_D);
1091 P 1214 1 OPERATOR_ROUTINE (G, G, G, SUB_G_G);
1092 P 1215 1 OPERATOR_ROUTINE (H, H, H, SUB_H_H);
1093 1216 1 OPERATOR_ROUTINE (P, P, P, SUB_P_P));
1094 1217 1
1095 1218 1
1096 1219 1 ! Define the Operator Routine Table for BASIC multiplication.
1097 1220 1 !
1098 P 1221 1 OPERATOR_ROUTINE_TABLE (BASIC_MUL_TABLE,
1099 P 1222 1 OPERATOR_ROUTINE (B, B, B, MUL_B_B);
1100 P 1223 1 OPERATOR_ROUTINE (W, W, W, MUL_W_W);
1101 P 1224 1 OPERATOR_ROUTINE (L, L, L, MUL_L_L);
1102 P 1225 1 OPERATOR_ROUTINE (F, F, F, MUL_F_F);
1103 P 1226 1 OPERATOR_ROUTINE (D, D, D, MUL_D_D);
1104 P 1227 1 OPERATOR_ROUTINE (G, G, G, MUL_G_G);
1105 P 1228 1 OPERATOR_ROUTINE (H, H, H, MUL_H_H);
1106 1229 1 OPERATOR_ROUTINE (P, P, P, MUL_P_P));
1107 1230 1
1108 1231 1
1109 1232 1 ! Define the Operator Routine Table for BASIC division.
1110 1233 1 !
1111 P 1234 1 OPERATOR_ROUTINE_TABLE (BASIC_DIV_TABLE,
1112 P 1235 1 OPERATOR_ROUTINE (B, B, B, DIV_B_B);
1113 P 1236 1 OPERATOR_ROUTINE (W, W, W, DIV_W_W);
1114 P 1237 1 OPERATOR_ROUTINE (L, L, L, DIV_L_L);
1115 P 1238 1 OPERATOR_ROUTINE (F, F, F, DIV_F_F);
1116 P 1239 1 OPERATOR_ROUTINE (D, D, D, DIV_D_D);
1117 P 1240 1 OPERATOR_ROUTINE (G, G, G, DIV_G_G);
1118 P 1241 1 OPERATOR_ROUTINE (H, H, H, DIV_H_H);
1119 1242 1 OPERATOR_ROUTINE (P, P, P, DIV_P_P));
1120 1243 1
1121 1244 1
1122 1245 1 ! Define the Operator Routine Table for BASIC unary plus.
1123 1246 1 !
1124 P 1247 1 OPERATOR_ROUTINE_TABLE (BASIC_UNARY_PLUS_TABLE,
1125 P 1248 1
1126 P 1249 1 ! The following are not language dependent types. This is needed for DEBUG
1127 P 1250 1 ! types. For example, DEP/QUAD L= +1.
1128 P 1251 1 !
```

```
1129 P 1252 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q);
1130 P 1253 1 OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O);
1131 P 1254 1
1132 P 1255 1 OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B);
1133 P 1256 1 OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W);
1134 P 1257 1 OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L);
1135 P 1258 1 OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F);
1136 P 1259 1 OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D);
1137 P 1260 1 OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G);
1138 P 1261 1 OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H);
1139 P 1262 1 OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P);
1140 1263 1
1141 1264 1
1142 1265 1 ! Define the Operator Routine Table for BASIC unary minus.
1143 1266 1
1144 P 1267 1 OPERATOR_ROUTINE_TABLE (BASIC_UNARY_MINUS_TABLE,
1145 P 1268 1 ! The following are not language dependent types. This is needed for DEBUG
1146 P 1269 1 ! types. For example, DEP/QUAD L= +1.
1147 P 1270 1
1148 P 1271 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q);
1149 P 1272 1 OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O);
1150 P 1273 1
1151 P 1274 1 OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B);
1152 P 1275 1 OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W);
1153 P 1276 1 OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L);
1154 P 1277 1 OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F);
1155 P 1278 1 OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D);
1156 P 1279 1 OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G);
1157 P 1280 1 OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H);
1158 P 1281 1 OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P);
1159 1282 1
1160 1283 1
1161 1284 1 ! Define the Operator Routine Table for BASIC exponentiation.
1162 1285 1 ! Exponentiation has some mixed forms. For example, if you raise a
1163 1286 1 ! floating number to an integer power, you do not necessarily
1164 1287 1 ! want to first convert the int to float. Instead, a special
1165 1288 1 ! routine indices such as POWER_F_L are provided to do the right thing here.
1166 1289 1
1167 P 1290 1 OPERATOR_ROUTINE_TABLE (BASIC_POWER_TABLE,
1168 P 1291 1 OPERATOR_ROUTINE (W, W, W, POWER_W_W);
1169 P 1292 1 OPERATOR_ROUTINE (L, L, L, POWER_L_L);
1170 P 1293 1 OPERATOR_ROUTINE (F, L, F, POWER_F_L);
1171 P 1294 1 OPERATOR_ROUTINE (D, L, D, POWER_D_L);
1172 P 1295 1 OPERATOR_ROUTINE (G, L, G, POWER_G_L);
1173 P 1296 1 OPERATOR_ROUTINE (H, L, H, POWER_H_L);
1174 P 1297 1 OPERATOR_ROUTINE (F, F, F, POWER_F_F);
1175 P 1298 1 OPERATOR_ROUTINE (D, F, D, POWER_D_F);
1176 P 1299 1 OPERATOR_ROUTINE (F, D, D, POWER_F_D);
1177 P 1300 1 OPERATOR_ROUTINE (D, D, D, POWER_D_D);
1178 P 1301 1 OPERATOR_ROUTINE (G, G, G, POWER_G_G);
1179 P 1302 1 OPERATOR_ROUTINE (H, H, H, POWER_H_H);
1180 1303 1
1181 1304 1
1182 1305 1 ! Define the Operator Routine Table for BASIC equal.
1183 1306 1 ! This operator can be done on strings as well as all the numeric
1184 1307 1 ! types.
1185 1308 1
```

```
1186 P 1309 1 OPERATOR ROUTINE TABLE (BASIC EQL TABLE,
1187 P 1310 1 OPERATOR_ROUTINE (RFA, RFA, L, EQL_RFA_RFA),
1188 P 1311 1 OPERATOR_ROUTINE (T, T, L, EQL_T_T),
1189 P 1312 1 OPERATOR_ROUTINE (B, B, L, EQL_B_B),
1190 P 1313 1 OPERATOR_ROUTINE (W, W, L, EQL_W_W),
1191 P 1314 1 OPERATOR_ROUTINE (L, L, L, EQL_L_L),
1192 P 1315 1 OPERATOR_ROUTINE (F, F, L, EQL_F_F),
1193 P 1316 1 OPERATOR_ROUTINE (D, D, L, EQL_D_D),
1194 P 1317 1 OPERATOR_ROUTINE (G, G, L, EQL_G_G),
1195 P 1318 1 OPERATOR_ROUTINE (H, H, L, EQL_H_H),
1196 P 1319 1 OPERATOR_ROUTINE (P, P, L, EQL_P_P);
1197 1320
1198 1321
1199 1322 1 Define the Operator Routine Table for BASIC not equal.
1200 1323 1 This operator can be done on strings as well as all the numeric types.
1201 1324
1202 P 1325 1 OPERATOR ROUTINE TABLE (BASIC NEQ TABLE,
1203 P 1326 1 OPERATOR_ROUTINE (RFA, RFA, L, NEQ_RFA_RFA),
1204 P 1327 1 OPERATOR_ROUTINE (T, T, L, NEQ_T_T),
1205 P 1328 1 OPERATOR_ROUTINE (B, B, L, NEQ_B_B),
1206 P 1329 1 OPERATOR_ROUTINE (W, W, L, NEQ_W_W),
1207 P 1330 1 OPERATOR_ROUTINE (L, L, L, NEQ_L_L),
1208 P 1331 1 OPERATOR_ROUTINE (F, F, L, NEQ_F_F),
1209 P 1332 1 OPERATOR_ROUTINE (D, D, L, NEQ_D_D),
1210 P 1333 1 OPERATOR_ROUTINE (G, G, L, NEQ_G_G),
1211 P 1334 1 OPERATOR_ROUTINE (H, H, L, NEQ_H_H),
1212 P 1335 1 OPERATOR_ROUTINE (P, P, L, NEQ_P_P);
1213 1336
1214 1337 1 In the tables for the comparison operators,
1215 1338 1 we allow strings to be compared, and also all the numeric types.
1216 1339 1 Define the Operator Routine Table for BASIC greater than.
1217 1340
1218 P 1341 1 OPERATOR ROUTINE TABLE (BASIC GTR TABLE,
1219 P 1342 1 OPERATOR_ROUTINE (T, T, L, GTR_T_T),
1220 P 1343 1 OPERATOR_ROUTINE (B, B, L, GTR_B_B),
1221 P 1344 1 OPERATOR_ROUTINE (W, W, L, GTR_W_W),
1222 P 1345 1 OPERATOR_ROUTINE (L, L, L, GTR_L_L),
1223 P 1346 1 OPERATOR_ROUTINE (F, F, L, GTR_F_F),
1224 P 1347 1 OPERATOR_ROUTINE (D, D, L, GTR_D_D),
1225 P 1348 1 OPERATOR_ROUTINE (G, G, L, GTR_G_G),
1226 P 1349 1 OPERATOR_ROUTINE (H, H, L, GTR_H_H),
1227 P 1350 1 OPERATOR_ROUTINE (P, P, L, GTR_P_P);
1228 1351
1229 1352
1230 1353 1 Define the Operator Routine Table for BASIC greater than or equal to.
1231 1354
1232 P 1355 1 OPERATOR ROUTINE TABLE (BASIC GEQ TABLE,
1233 P 1356 1 OPERATOR_ROUTINE (T, T, L, GEQ_T_T),
1234 P 1357 1 OPERATOR_ROUTINE (B, B, L, GEQ_B_B),
1235 P 1358 1 OPERATOR_ROUTINE (W, W, L, GEQ_W_W),
1236 P 1359 1 OPERATOR_ROUTINE (L, L, L, GEQ_L_L),
1237 P 1360 1 OPERATOR_ROUTINE (F, F, L, GEQ_F_F),
1238 P 1361 1 OPERATOR_ROUTINE (D, D, L, GEQ_D_D),
1239 P 1362 1 OPERATOR_ROUTINE (G, G, L, GEQ_G_G),
1240 P 1363 1 OPERATOR_ROUTINE (H, H, L, GEQ_H_H),
1241 P 1364 1 OPERATOR_ROUTINE (P, P, L, GEQ_P_P);
1242 1365
```

```
1243      1366 1
1244      1367 1 ! Define the Operator Routine Table for BASIC less than.
1245      1368 1
1246      P 1369 1 OPERATOR_ROUTINE_TABLE (BASIC_LSS_TABLE,
1247      P 1370 1   OPERATOR_ROUTINE (T, T, L, LSS_T_T),
1248      P 1371 1   OPERATOR_ROUTINE (B, B, L, LSS_B_B),
1249      P 1372 1   OPERATOR_ROUTINE (W, W, L, LSS_W_W),
1250      P 1373 1   OPERATOR_ROUTINE (L, L, L, LSS_L_L),
1251      P 1374 1   OPERATOR_ROUTINE (F, F, L, LSS_F_F),
1252      P 1375 1   OPERATOR_ROUTINE (D, D, L, LSS_D_D),
1253      P 1376 1   OPERATOR_ROUTINE (G, G, L, LSS_G_G),
1254      P 1377 1   OPERATOR_ROUTINE (H, H, L, LSS_H_H),
1255      1378 1   OPERATOR_ROUTINE (P, P, L, LSS_P_P));
1256      1379 1
1257      1380 1
1258      1381 1 ! Define the Operator Routine Table for BASIC less than or equal to.
1259      1382 1
1260      P 1383 1 OPERATOR_ROUTINE_TABLE (BASIC_LEQ_TABLE,
1261      P 1384 1   OPERATOR_ROUTINE (T, T, L, LEQ_T_T),
1262      P 1385 1   OPERATOR_ROUTINE (B, B, L, LEQ_B_B),
1263      P 1386 1   OPERATOR_ROUTINE (W, W, L, LEQ_W_W),
1264      P 1387 1   OPERATOR_ROUTINE (L, L, L, LEQ_L_L),
1265      P 1388 1   OPERATOR_ROUTINE (F, F, L, LEQ_F_F),
1266      P 1389 1   OPERATOR_ROUTINE (D, D, L, LEQ_D_D),
1267      P 1390 1   OPERATOR_ROUTINE (G, G, L, LEQ_G_G),
1268      P 1391 1   OPERATOR_ROUTINE (H, H, L, LEQ_H_H),
1269      1392 1   OPERATOR_ROUTINE (P, P, L, LEQ_P_P));
1270      1393 1
1271      1394 1
1272      1395 1 ! The logical operators .AND., .OR., .EQV., .NEQV., .NOT. can be applied
1273      1396 1 ! only to integer data types.
1274      1397 1
1275      1398 1 ! Define the Operator Routine Table for BASIC not.
1276      1399 1
1277      P 1400 1 OPERATOR_ROUTINE_TABLE (BASIC_BIT_NOT_TABLE,
1278      P 1401 1   OPERATOR_ROUTINE (B, B, B, BIT_NOT_B),
1279      P 1402 1   OPERATOR_ROUTINE (W, W, W, BIT_NOT_W),
1280      1403 1   OPERATOR_ROUTINE (L, L, L, BIT_NOT_L));
1281      1404 1
1282      1405 1
1283      1406 1 ! Define the Operator Routine Table for BASIC and.
1284      1407 1
1285      P 1408 1 OPERATOR_ROUTINE_TABLE (BASIC_BIT_AND_TABLE,
1286      P 1409 1   OPERATOR_ROUTINE (B, B, B, BIT_AND_B_B),
1287      P 1410 1   OPERATOR_ROUTINE (W, W, W, BIT_AND_W_W),
1288      1411 1   OPERATOR_ROUTINE (L, L, L, BIT_AND_L_L));
1289      1412 1
1290      1413 1
1291      1414 1 ! Define the Operator Routine Table for BASIC or.
1292      1415 1
1293      P 1416 1 OPERATOR_ROUTINE_TABLE (BASIC_BIT_OR_TABLE,
1294      P 1417 1   OPERATOR_ROUTINE (B, B, B, BIT_OR_B_B),
1295      P 1418 1   OPERATOR_ROUTINE (W, W, W, BIT_OR_W_W),
1296      1419 1   OPERATOR_ROUTINE (L, L, L, BIT_OR_L_L));
1297      1420 1
1298      1421 1
1299      1422 1 ! Define the Operator Routine Table for BASIC xor, neqv
```

```
1300      1423 1 !
1301      1424 1 OPERATOR_ROUTINE_TABLE (BASIC_BIT_XOR_TABLE,
1302      1425 1   OPERATOR_ROUTINE (B, B, B, BIT_XOR_B_B),
1303      1426 1   OPERATOR_ROUTINE (W, W, W, BIT_XOR_W_W),
1304      1427 1   OPERATOR_ROUTINE (L, L, L, BIT_XOR_L_L));
1305      1428 1
1306      1429 1
1307      1430 1 ! Define the Operator Routine Table for BASIC eqv.
1308      1431 1
1309      1432 1 OPERATOR_ROUTINE_TABLE (BASIC_BIT_EQV_TABLE,
1310      1433 1   OPERATOR_ROUTINE (B, B, B, BIT_EQV_B_B),
1311      1434 1   OPERATOR_ROUTINE (W, W, W, BIT_EQV_W_W),
1312      1435 1   OPERATOR_ROUTINE (L, L, L, BIT_EQV_L_L));
1313      1436 1
1314      1437 1
1315      1438 1 ! Define the Operator Routine Table for BASIC imp.
1316      1439 1
1317      1440 1 OPERATOR_ROUTINE_TABLE (BASIC_BIT_IMP_TABLE,
1318      1441 1   OPERATOR_ROUTINE (B, B, B, BIT_IMP_B_B),
1319      1442 1   OPERATOR_ROUTINE (W, W, W, BIT_IMP_W_W),
1320      1443 1   OPERATOR_ROUTINE (L, L, L, BIT_IMP_L_L));
1321      1444 1
1322      1445 1
1323      1446 1 ! Define the Operator Information Table for BASIC.
1324      1447 1
1325      1448 1 OPERATOR_INFO_TABLE (BASIC_OPINFO_TABLE,
1326      1449 1
1327      1450 1   ! The following are arithmetic tables that accept all numeric data types,
1328      1451 1   ! including complex. They thus go through the larger HIER1 table, and
1329      1452 1   ! need to specify an incompatibility table.
1330      1453 1
1331      1454 1   OPERATOR_INFO_ENTRY (ADD, BASIC_ADD_TABLE, BASIC_HIER1_TABLE,
1332      1455 1   TABLEBASET,
1333      1456 1   OPERATOR_INFO_ENTRY (SUBTRACT, BASIC_SUB_TABLE, BASIC_HIER1_TABLE,
1334      1457 1   TABLEBASET,
1335      1458 1   OPERATOR_INFO_ENTRY (MULTIPLY, BASIC_MUL_TABLE, BASIC_HIER1_TABLE,
1336      1459 1   TABLEBASET,
1337      1460 1   OPERATOR_INFO_ENTRY (DIVIDE, BASIC_DIV_TABLE, BASIC_HIER1_TABLE,
1338      1461 1   TABLEBASET,
1339      1462 1   OPERATOR_INFO_ENTRY (UNARY_PLUS, BASIC_UNARY_PLUS_TABLE,
1340      1463 1   BASIC_HIER1_TABLE, TABLEBASET),
1341      1464 1   OPERATOR_INFO_ENTRY (UNARY_MINUS, BASIC_UNARY_MINUS_TABLE,
1342      1465 1   BASIC_HIER1_TABLE, TABLEBASET),
1343      1466 1   OPERATOR_INFO_ENTRY (POWER_OF, BASIC_POWER_TABLE, BASIC_HIER1_TABLE,
1344      1467 1   TABLEBASET,
1345      1468 1
1346      1469 1   ! The relationals accept all numeric types and thus need the larger
1347      1470 1   ! hierarchy table. There is no incompatibility table.
1348      1471 1
1349      1472 1   OPERATOR_INFO_ENTRY (EQUAL, BASIC_EQL_TABLE, BASIC_HIER1_TABLE,
1350      1473 1   TABLEBASET,
1351      1474 1   OPERATOR_INFO_ENTRY (NOT_EQUAL, BASIC_NEQ_TABLE, BASIC_HIER1_TABLE,
1352      1475 1   TABLEBASET,
1353      1476 1   OPERATOR_INFO_ENTRY (GTR_THAN, BASIC_GTR_TABLE, BASIC_HIER1_TABLE,
1354      1477 1   TABLEBASET,
1355      1478 1   OPERATOR_INFO_ENTRY (GTR_EQUAL, BASIC_GEQ_TABLE, BASIC_HIER1_TABLE,
1356      1479 1   TABLEBASET),
```

1357 P 1480 1  
1358 P 1481 1  
1359 P 1482 1  
1360 P 1483 1  
1361 P 1484 1  
1362 P 1485 1  
1363 P 1486 1  
1364 P 1487 1  
1365 P 1488 1  
1366 P 1489 1  
1367 P 1490 1  
1368 P 1491 1  
1369 P 1492 1  
1370 P 1493 1  
1371 P 1494 1  
1372 P 1495 1  
1373 P 1496 1  
1374 P 1497 1  
1375 P 1498 1  
1376 P 1499 1  
1377 P 1500 1  
1378 P 1501 1  
1379 P 1502 1  
1380 P 1503 1  
1381 P 1504 1  
1382 P 1505 1  
1383 P 1506 1  
1384 P 1507 1  
1385 P 1508 1  
1386 P 1509 1  
1387 P 1510 1  
1388 P 1511 1  
1389 P 1512 1  
1390 P 1513 1  
1391 P 1514 1  
1392 P 1515 1  
1393 P 1516 1  
1394 P 1517 1  
1395 P 1518 1  
1396 P 1519 1  
1397 P 1520 1  
1398 P 1521 1  
1399 P 1522 1  
1400 P 1523 1  
1401 P 1524 1  
1402 P 1525 1  
1403 P 1526 1  
1404 P 1527 1  
1405 P 1528 1  
1406 P 1529 1

OPERATOR INFO ENTRY (LSS\_THAN, BASIC\_LSS\_TABLE, BASIC\_HIER1\_TABLE,  
TABLEBASE),  
OPERATOR INFO ENTRY (LSS\_EQUAL, BASIC\_LEQ\_TABLE, BASIC\_HIER1\_TABLE,  
TABLEBASE),

! The logical operators accept only integer quantities so they can  
! use the smallest hierarchy table. They also do not need an  
! incompatibility table.

OPERATOR INFO ENTRY (BIT\_NOT, BASIC\_BIT\_NOT\_TABLE, BASIC\_HIER2\_TABLE,  
TABLEBASE),  
OPERATOR INFO ENTRY (BIT\_AND, BASIC\_BIT\_AND\_TABLE, BASIC\_HIER2\_TABLE,  
TABLEBASE),  
OPERATOR INFO ENTRY (BIT\_OR, BASIC\_BIT\_OR\_TABLE, BASIC\_HIER2\_TABLE,  
TABLEBASE),  
OPERATOR INFO ENTRY (BIT\_XOR, BASIC\_BIT\_XOR\_TABLE, BASIC\_HIER2\_TABLE,  
TABLEBASE),  
OPERATOR INFO ENTRY (BIT\_EQV, BASIC\_BIT\_EQV\_TABLE, BASIC\_HIER2\_TABLE,  
TABLEBASE),  
OPERATOR INFO ENTRY (BIT\_IMP, BASIC\_BIT\_IMP\_TABLE, BASIC\_HIER2\_TABLE,  
TABLEBASE),

! The CONVERT operator gets called to convert subscripts to integer  
! type and to convert expressions in FOR, IF, WHILE, REPEAT statements  
! to integer type. It can thus use the smaller HIER2 table to specify  
! the rules for conversion to integer.

OPERATOR INFO ENTRY (CONVERT, TABLEBASE, BASIC\_HIER2\_TABLE,  
TABLEBASE),

! The DEPOSIT operator gets called on the DEPOSIT command. It has  
! its own hierarchy table which allows any numeric type to be  
! converted to any other numeric type. The incompatibility table,  
! however, still prevents depositing D types into G types and  
! vice versa.

OPERATOR INFO ENTRY (DEPOSIT, TABLEBASE, BASIC\_HIERD\_TABLE,  
TABLEBASE),

! The IDENTITY operator gets called at the end of an EVALUATE command  
! to apply the PRIM\_TO\_VAL routine and then apply the appropriate  
! type mappings. This will ensure that EV BU will print as a signed integer,  
! for example.

! The identity operator does not require any tables.

OPERATOR\_INFO\_ENTRY (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE)  
);

```
1408 1530 1
1409 1531 1
1410 1532 1
1411 1533 1
1412 1534 1
1413 1535 1
1414 1536 1
1415 1537 1
1416 1538 1
1417 1539 1
1418 1540 1
1419 1541 1
1420 1542 1
1421 1543 1
1422 1544 1
1423 1545 1
1424 1546 1
1425 1547 1
1426 1548 1
1427 1549 1
1428 1550 1
1429 1551 1
1430 1552 1
1431 1553 1
1432 1554 1
1433 1555 1
1434 1556 1
1435 1557 1
1436 1558 1
1437 1559 1
1438 1560 1
1439 1561 1
1440 1562 1
1441 1563 1
1442 1564 1
1443 1565 1
1444 P 1566 1
1445 1567 1
1446 1568 1
1447 1569 1
1448 1570 1
1449 1571 1
1450 P 1572 1
1451 1573 1
1452 1574 1
1453 1575 1
1454 1576 1
1455 1577 1
1456 1578 1
1457 P 1579 1
1458 P 1580 1
1459 P 1581 1
1460 P 1582 1
1461 P 1583 1
1462 P 1584 1
1463 1585 1
1464 1586 1

BLISS OPERATOR INFORMATION TABLES

This section contains the Operator Routine and Type tables needed to
evaluate expressions in BLISS.

Here is the list of types that we may get back from DBG$PRIM_TO_VAL
when we pass it a BLISS primary:

B      - byte signed
BU     - byte unsigned
W      - word signed
WU     - word unsigned
L      - longword signed
LU     - longword unsigned
V      - aligned bitstring. e.g., the primary X<0,5,0>
SV     - signed aligned bitstring. e.g., X<0,5,1>
VU     - unaligned bitstring. e.g., X<1,5,0>
SVU    - signed unaligned bitstring. e.g., X<1,5,1>
PTR    - we get this back for REF items
Z      - "unknown" - we get this back for BLISS field names
        e.g., X[fieldname]. No operations are allowed on
        fieldnames so this type does not appear in the
        tables.
ZI     - instruction, XLINE 10
ZEM    - entry mask, e.g., EV ROUT_NAME

DBG$PRIM_TO_ADDR always returns type L.

Define a table that maps PTR type into integer. Type PTR can come
back from REF objects.

TYPE MAPPING TABLE (BLISS_MAP_TABLE,
TYPE_GRAPH_EDGE (PTR,[]));

Define the Type Conversion Information Table for BLISS.
This points to the above mapping table.

CONVERSION INFO TABLE (BLISS_CVTINFO_TABLE,
CONVERSION_INFO_ENTRY (BLISS_MAP_TABLE, TABLEBASE));

Define the Type Hierarchy Table. All operations are done on signed longwords.
Thus we provide a path for all types to be converted to signed longwords.

TYPE HIERARCHY TABLE (BLISS_HIER_TABLE,
TYPE_GRAPH_EDGE (BU, WU),
TYPE_GRAPH_EDGE (WU, LU),
TYPE_GRAPH_EDGE (LU, L),
TYPE_GRAPH_EDGE (B, W),
TYPE_GRAPH_EDGE (W, L),
0);
```

```
1465 1587 1
1466 1588 1
1467 1589 1
1468 1590 1
1469 1591 1
1470 1592 1
1471 1593 1
1472 1594 1
1473 1595 1
1474 1596 1
1475 1597 1
1476 1598 1
1477 1599 1
1478 1600 1
1479 1601 1
1480 1602 1
1481 1603 1
1482 1604 1
1483 1605 1
1484 1606 1
1485 1607 1
1486 1608 1
1487 1609 1
1488 1610 1
1489 1611 1
1490 1612 1
1491 1613 1
1492 1614 1
1493 1615 1
1494 1616 1
1495 1617 1
1496 1618 1
1497 1619 1
1498 1620 1
1499 1621 1
1500 1622 1
1501 1623 1
1502 1624 1
1503 1625 1
1504 1626 1
1505 1627 1
1506 1628 1
1507 1629 1
1508 1630 1
1509 1631 1
1510 1632 1
1511 1633 1
1512 1634 1
1513 1635 1
1514 1636 1
1515 1637 1
1516 1638 1
1517 1639 1
1518 1640 1
1519 1641 1
1520 1642 1
1521 1643 1

P 1593 1
P 1594 1
P 1595 1
P 1596 1
P 1597 1
P 1598 1
P 1599 1
P 1600 1
P 1601 1
P 1602 1
P 1603 1
P 1604 1
P 1605 1
P 1606 1
P 1607 1
P 1608 1
P 1609 1
P 1610 1
P 1611 1
P 1612 1
P 1613 1
P 1614 1
P 1615 1
P 1616 1
P 1617 1
P 1618 1
P 1619 1
P 1620 1
P 1621 1
P 1622 1
P 1623 1
P 1624 1
P 1625 1
P 1626 1
P 1627 1
P 1628 1
P 1629 1
P 1630 1
P 1631 1
P 1632 1
P 1633 1
P 1634 1
P 1635 1
P 1636 1
P 1637 1
P 1638 1
P 1639 1
P 1640 1
P 1641 1
P 1642 1
P 1643 1

!! Define the Type Hierarchy Table for DEPOSIT.
!! This is a circular graph which includes all types that can be obtained
!! from calling PRIM_TO_VAL on a BLISS primary. What this means is that
!! any type is convertible to any other type on a DEPOSIT.
TYPE_HIERARCHY_TABLE (BLISS_HIERD_TABLE,
  TYPE_GRAPH_EDGE (B, BU);
  TYPE_GRAPH_EDGE (BU, W);
  TYPE_GRAPH_EDGE (W, WU);
  TYPE_GRAPH_EDGE (WU, L);
  TYPE_GRAPH_EDGE (L, LU);
  TYPE_GRAPH_EDGE (LU, VU);
  TYPE_GRAPH_EDGE (VU, SVU);
  TYPE_GRAPH_EDGE (SVU, SV);
  TYPE_GRAPH_EDGE (SV, V);
  TYPE_GRAPH_EDGE (V, B);
  0);

++
!! All of the arithmetic operations below are defined to work on signed
!! longwords.
!! Note that only the fetch operator (.) does an implicit fetch for BLISS.
!! This means, for example, that
!! EVAL A+B
!! will add the addresses of A and B, not the values. Addresses are obtained
!! from the DBG$PRIM_TO_ADDR, and their type is always L.
!! If the fetch is done, the DBG$PRIM_TO_VAL routine gets called, and the value
!! is pulled from that value descriptor and put into the result value
!! descriptor, which is of type longword. Thus the result of the fetch
!! operator is always longword. Any extraction of byte, word, bitfield, etc.
!! values is done inside of DBG$PRIM_TO_VAL.
!! For example,
!! EVAL .W + .BU ! W is word signed, BU is byte unsigned
!! The .W operator is first applied. DBG$PRIM_TO_VAL is called and it
!! extracts the word quantity for W, sign extends it to a longword,
!! and places the longword value in the Value Descriptor. We copy that
!! value into the Value Descriptor containing the result of the .W operation.
!! Similarly we obtain a type L descriptor with the value of BU, properly
!! zero-extended. These are then added as longwords, and the result is
!! a longword.
!! So the upshot of all this is that none of the operations except FETCH
!! ever see anything except dtype L.
!!
!! Define the Operator Routine Table for BLISS addition.
OPERATOR_ROUTINE_TABLE (BLISS_ADD_TABLE,
  OPERATOR_ROUTINE (L, L, L; ADD_L_L));
!! Define the Operator Routine Table for BLISS subtraction.
OPERATOR_ROUTINE_TABLE (BLISS_SUB_TABLE,
```

```
1522 1644 1 OPERATOR_ROUTINE (L, L, L, SUB_L_L));
1523 1645 1
1524 1646 1 ! Define the Operator Routine Table for BLISS Multiplication.
1525 1647 1
1526 P 1648 1 OPERATOR_ROUTINE_TABLE (BLISS_MUL_TABLE,
1527 1649 1 OPERATOR_ROUTINE (L, L, L, MUL_L_L));
1528 1650 1
1529 1651 1 ! Define the Operator Routine Table for BLISS Division.
1530 1652 1
1531 P 1653 1 OPERATOR_ROUTINE_TABLE (BLISS_DIV_TABLE,
1532 1654 1 OPERATOR_ROUTINE (L, L, L, DIV_L_L));
1533 1655 1
1534 1656 1 ! Define the Operator Routine Table for BLISS Modulus.
1535 1657 1 ! The BLISS modulus function is actually a remainder function.
1536 1658 1
1537 P 1659 1 OPERATOR_ROUTINE_TABLE (BLISS_MOD_TABLE,
1538 1660 1 OPERATOR_ROUTINE (L, L, L, REM_L_L));
1539 1661 1
1540 1662 1 ! Define the Operator Routine Table for BLISS arithmetic shift.
1541 1663 1 ! There is only one shift operation in BLISS. It uses the "A" symbol.
1542 1664 1 ! A positive right argument indicates left shift and a negative right
1543 1665 1 ! argument indicates right shift. This is how the SHIFT_LEFT_L_L routine
1544 1666 1 ! behaves.
1545 1667 1
1546 P 1668 1 OPERATOR_ROUTINE_TABLE (BLISS_SHIFT_TABLE,
1547 1669 1 OPERATOR_ROUTINE (L, L, L, SHIFT_LEFT_L_L));
1548 1670 1
1549 1671 1 ! Define the Operator Routine Table for BLISS Equal.
1550 1672 1
1551 P 1673 1 OPERATOR_ROUTINE_TABLE (BLISS_EQUAL_TABLE,
1552 1674 1 OPERATOR_ROUTINE (L, L, L, EQL_L_L));
1553 1675 1
1554 1676 1 ! Define the Operator Routine Table for BLISS Not Equal.
1555 1677 1
1556 P 1678 1 OPERATOR_ROUTINE_TABLE (BLISS_NOT_EQUAL_TABLE,
1557 1679 1 OPERATOR_ROUTINE (L, L, L, NEQ_L_L));
1558 1680 1
1559 1681 1 ! Define the Operator Routine Table for BLISS Less Than.
1560 1682 1
1561 P 1683 1 OPERATOR_ROUTINE_TABLE (BLISS_LSS_THAN_TABLE,
1562 1684 1 OPERATOR_ROUTINE (L, L, L, LSS_L_L));
1563 1685 1
1564 1686 1 ! Define the Operator Routine Table for BLISS Less Than Unsigned.
1565 1687 1
1566 P 1688 1 OPERATOR_ROUTINE_TABLE (BLISS_LSSU_THAN_TABLE,
1567 1689 1 OPERATOR_ROUTINE (L, L, L, LSS_LU_LO));
1568 1690 1
1569 1691 1 ! Define the Operator Routine Table for BLISS Greater Than.
1570 1692 1
1571 P 1693 1 OPERATOR_ROUTINE_TABLE (BLISS_GTR_THAN_TABLE,
1572 1694 1 OPERATOR_ROUTINE (L, L, L, GTR_L_L));
1573 1695 1
1574 1696 1 ! Define the Operator Routine Table for BLISS Greater Than Unsigned.
1575 1697 1
1576 P 1698 1 OPERATOR_ROUTINE_TABLE (BLISS_GTRU_THAN_TABLE,
1577 1699 1 OPERATOR_ROUTINE (L, L, L, GTR_LU_LO));
1578 1700 1
```

```
1579 1701 1 ! Define the Operator Routine Table for BLISS Less Than or Equal.
1580 1702 1
1581 P 1703 1 OPERATOR_ROUTINE_TABLE (BLISS_LSS_EQUAL_TABLE,
1582 1704 1 OPERATOR_ROUTINE (L, L, L, LEQ_L_L));
1583 1705 1
1584 1706 1 ! Define the Operator Routine Table for BLISS Less Than or Equal Unsigned.
1585 1707 1
1586 P 1708 1 OPERATOR_ROUTINE_TABLE (BLISS_LSSU_EQUAL_TABLE,
1587 1709 1 OPERATOR_ROUTINE (L, L, L, LEQ_LU_LU));
1588 1710 1
1589 1711 1 ! Define the Operator Routine Table for BLISS Greater Than or Equal.
1590 1712 1
1591 P 1713 1 OPERATOR_ROUTINE_TABLE (BLISS_GTR_EQUAL_TABLE,
1592 1714 1 OPERATOR_ROUTINE (L, L, L, GEQ_L_L));
1593 1715 1
1594 1716 1 ! Define the Operator Routine Table for BLISS Greater Than or Equal Unsigned.
1595 1717 1
1596 P 1718 1 OPERATOR_ROUTINE_TABLE (BLISS_GTRU_EQUAL_TABLE,
1597 1719 1 OPERATOR_ROUTINE (L, L, L, GEQ_LU_LU));
1598 1720 1
1599 1721 1 ! Define the Operator Routine Table for BLISS Bitwise And.
1600 1722 1
1601 P 1723 1 OPERATOR_ROUTINE_TABLE (BLISS_BIT_AND_TABLE,
1602 1724 1 OPERATOR_ROUTINE (L, L, L, BIT_AND_L_L));
1603 1725 1
1604 1726 1 ! Define the Operator Routine Table for BLISS Bitwise Or.
1605 1727 1
1606 P 1728 1 OPERATOR_ROUTINE_TABLE (BLISS_BIT_OR_TABLE,
1607 1729 1 OPERATOR_ROUTINE (L, L, L, BIT_OR_L_L));
1608 1730 1
1609 1731 1 ! Define the Operator Routine Table for BLISS Bitwise Xor.
1610 1732 1
1611 P 1733 1 OPERATOR_ROUTINE_TABLE (BLISS_BIT_XOR_TABLE,
1612 1734 1 OPERATOR_ROUTINE (L, L, L, BIT_XOR_L_L));
1613 1735 1
1614 1736 1 ! Define the Operator Routine Table for BLISS Bitwise Eqv.
1615 1737 1
1616 P 1738 1 OPERATOR_ROUTINE_TABLE (BLISS_BIT_EQV_TABLE,
1617 1739 1 OPERATOR_ROUTINE (L, L, L, BIT_EQV_L_L));
1618 1740 1
1619 1741 1 ! Define the Operator Routine Table for BLISS Unary Plus.
1620 1742 1
1621 P 1743 1 OPERATOR_ROUTINE_TABLE (BLISS_UNARY_PLUS_TABLE,
1622 P 1744 1
1623 P 1745 1 ! The following are not language dependent types. This is needed for DEBUG
1624 P 1746 1 types. For example, DEP/QUAD L= +1.
1625 P 1747 1
1626 P 1748 1 OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
1627 P 1749 1 OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
1628 P 1750 1 OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
1629 P 1751 1 OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
1630 P 1752 1 OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
1631 P 1753 1 OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
1632 P 1754 1 OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
1633 P 1755 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
1634 P 1756 1 OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
1635 P 1757 1
```

```
1636 1758 1 OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L));
1637 1759 1
1638 1760 1 ! Define the Operator Routine Table for BLISS Unary Minus.
1639 1761 1
1640 P 1762 1 OPERATOR_ROUTINE_TABLE (BLISS_UNARY_MINUS_TABLE,
1641 P 1763 1
1642 P 1764 1 ! The following are not language dependent types. This is needed for DEBUG
1643 P 1765 1 ! types. For example, DEP/QUAD L= +1.
1644 P 1766 1
1645 P 1767 1 OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
1646 P 1768 1 OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
1647 P 1769 1 OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
1648 P 1770 1 OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
1649 P 1771 1 OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
1650 P 1772 1 OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
1651 P 1773 1 OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
1652 P 1774 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
1653 P 1775 1 OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
1654 P 1776 1
1655 1777 1 OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L));
1656 1778 1
1657 1779 1 ! Define the Operator Routine Table for BLISS Bitwise Not.
1658 1780 1
1659 P 1781 1 OPERATOR_ROUTINE_TABLE (BLISS_BIT_NOT_TABLE,
1660 1782 1 OPERATOR_ROUTINE (L, L, L, BIT_NOT_L));
1661 1783 1
1662 1784 1 ! Define the Operator Routine Table for BLISS bit selection.
1663 1785 1 ! The bit-select operator X<p,s,e> can be applied to any
1664 1786 1 ! BLISS expression. It goes through the DBGSBLISS_BITSELECT
1665 1787 1 ! routine in DBGEVALOP.
1666 1788 1
1667 1789 1 ! The <p,s,e> operation can be viewed as just modifying the
1668 1790 1 ! address given by X. That is, X can be any BLISS expression,
1669 1791 1 ! and the result of all BLISS expressions is a longword.
1670 1792 1 ! If the X<p,s,e> expression does not have a fetch associated with
1671 1793 1 ! it, the value of X<p,s,e> is just X+p/8
1672 1794 1
1673 1795 1 ! If there is a fetch, then what happens is:
1674 1796 1 ! (1) The <p,s,e> operator is done first. The DBGSBLISS_BITSELECT routine just
1675 1797 1 ! squires away the information inside of DBGEVALOP.
1676 1798 1 ! Nothing is done with it until the fetch.
1677 1799 1 ! (2) the extraction of the BLISS field is
1678 1800 1 ! done at the evaluation of the fetch operator. (DBGSBLISS_INDIRECTION
1679 1801 1 ! in the DBGEVALOP module.)
1680 1802 1
1681 P 1803 1 OPERATOR_ROUTINE_TABLE (BLISS_BITSELECT_TABLE,
1682 1804 1 OPERATOR_ROUTINE (L, L, L, BITSELECT));
1683 1805 1
1684 1806 1 ! Define the Operator Routine Table for BLISS indirection.
1685 1807 1 ! Indirection is the only operator that call DBGSPRIM_TO_VAL to do
1686 1808 1 ! the implicit fetch, so it is the only one that may see all the possible
1687 1809 1 ! dtypes that we may get back from a BLISS primary. We thus include
1688 1810 1 ! those dtypes where it is legal to do a fetch. (E.g., .ROUT-NAME is not
1689 1811 1 ! legal, so that is not here).
1690 1812 1
1691 P 1813 1 OPERATOR_ROUTINE_TABLE (BLISS_INDIRECT_TABLE,
1692 P 1814 1 OPERATOR_ROUTINE (B, B, L, INDIRECT_LU),
```

```
1693 P 1815 1 OPERATOR_ROUTINE (W, W, L, INDIRECT_LU),
1694 P 1816 1 OPERATOR_ROUTINE (BU, BU, L, INDIRECT_LU),
1695 P 1817 1 OPERATOR_ROUTINE (WU, WU, L, INDIRECT_LU),
1696 P 1818 1 OPERATOR_ROUTINE (LU, LU, L, INDIRECT_LU),
1697 P 1819 1 OPERATOR_ROUTINE (V, V, L, INDIRECT_LU),
1698 P 1820 1 OPERATOR_ROUTINE (VU, VU, L, INDIRECT_LU),
1699 P 1821 1 OPERATOR_ROUTINE (SV, SV, L, INDIRECT_LU),
1700 P 1822 1 OPERATOR_ROUTINE (SVU, SVU, L, INDIRECT_LU),
1701 1823 1 OPERATOR_ROUTINE (L, L, L, INDIRECT_LU);
1702 1824 1
1703 1825 1
1704 1826 1 ! Define the Operator Information Table for BLISS.
1705 1827 1 !
1706 P 1828 1 OPERATOR_INFO_TABLE (BLISS_OPINFO_TABLE,
1707 P 1829 1
1708 P 1830 1 ! All of the arithmetic operators use the same hierarchy table,
1709 P 1831 1 ! BLISS_HIER_TABLE. They all have the fetch flag turned off, meaning
1710 P 1832 1 ! they do address arithmetic in the absence of an explicit fetch.
1711 P 1833 1 ! There is not incompatibility table for any of the BLISS operators.
1712 P 1834 1 !
1713 P 1835 1 OPERATOR_INFO_ENTRY
1714 P 1836 1 (UNARY_PLUS, BLISS_UNARY_PLUS_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1715 P 1837 1 OPERATOR_INFO_ENTRY
1716 P 1838 1 (UNARY_MINUS, BLISS_UNARY_MINUS_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1717 P 1839 1 OPERATOR_INFO_ENTRY
1718 P 1840 1 (BIT_NOT, BLISS_BIT_NOT_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1719 P 1841 1 OPERATOR_INFO_ENTRY
1720 P 1842 1 (ADD, BLISS_ADD_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1721 P 1843 1 OPERATOR_INFO_ENTRY
1722 P 1844 1 (SUBTRACT, BLISS_SUB_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1723 P 1845 1 OPERATOR_INFO_ENTRY
1724 P 1846 1 (MULTIPLY, BLISS_MUL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1725 P 1847 1 OPERATOR_INFO_ENTRY
1726 P 1848 1 (DIVIDE, BLISS_DIV_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1727 P 1849 1 OPERATOR_INFO_ENTRY
1728 P 1850 1 (REMAINDER, BLISS_MOD_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1729 P 1851 1 OPERATOR_INFO_ENTRY
1730 P 1852 1 (LEFT_SHIFT, BLISS_SHIFT_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1731 P 1853 1 OPERATOR_INFO_ENTRY
1732 P 1854 1 (EQUAL, BLISS_EQUAL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1733 P 1855 1 OPERATOR_INFO_ENTRY
1734 P 1856 1 (NOT_EQUAL, BLISS_NOT_EQUAL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1735 P 1857 1 OPERATOR_INFO_ENTRY
1736 P 1858 1 (GTR_THAN, BLISS_GTR_THAN_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1737 P 1859 1 OPERATOR_INFO_ENTRY
1738 P 1860 1 (GTR_THAN_U, BLISS_GTRU_THAN_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1739 P 1861 1 OPERATOR_INFO_ENTRY
1740 P 1862 1 (LSS_THAN, BLISS_LSS_THAN_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1741 P 1863 1 OPERATOR_INFO_ENTRY
1742 P 1864 1 (LSS_THAN_U, BLISS_LSSU_THAN_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1743 P 1865 1 OPERATOR_INFO_ENTRY
1744 P 1866 1 (GTR_EQUAL, BLISS_GTR_EQUAL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1745 P 1867 1 OPERATOR_INFO_ENTRY
1746 P 1868 1 (GTR_EQUAL_U, BLISS_GTRU_EQUAL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1747 P 1869 1 OPERATOR_INFO_ENTRY
1748 P 1870 1 (LSS_EQUAL, BLISS_LSS_EQUAL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
1749 P 1871 1 OPERATOR_INFO_ENTRY
```

```
1750 P 1872 1
1751 P 1873 1
1752 P 1874 1
1753 P 1875 1
1754 P 1876 1
1755 P 1877 1
1756 P 1878 1
1757 P 1879 1
1758 P 1880 1
1759 P 1881 1
1760 P 1882 1
1761 P 1883 1
1762 P 1884 1
1763 P 1885 1
1764 P 1886 1
1765 P 1887 1
1766 P 1888 1
1767 P 1889 1
1768 P 1890 1
1769 P 1891 1
1770 P 1892 1
1771 P 1893 1
1772 P 1894 1
1773 P 1895 1
1774 P 1896 1
1775 P 1897 1
1776 P 1898 1
1777 P 1899 1
1778 P 1900 1
1779 P 1901 1
1780 P 1902 1
1781 P 1903 1
1782 P 1904 1
1783 P 1905 1
1784 P 1906 1
1785 P 1907 1
1786 P 1908 1
1787 P 1909 1
1788 P 1910 1
1789 P 1911 1
1790 P 1912 1
1791 P 1913 1
1792 P 1914 1
1793 P 1915 1
1794 P 1916 1
1795 P 1917 1
1796 P 1918 1
1797 P 1919 1
1798 P 1920 1
1799 1921 1
1800 1922 1
```

```
(LSS_EQUAL U, BLISS_LSSU_EQUAL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
OPERATOR_INFO_ENTRY
(BIT_AND, BLISS_BIT_AND_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
OPERATOR_INFO_ENTRY
(BIT_OR, BLISS_BIT_OR_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
OPERATOR_INFO_ENTRY
(BIT_XOR, BLISS_BIT_XOR_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
OPERATOR_INFO_ENTRY
(BIT_EQV, BLISS_BIT_EQV_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
OPERATOR_INFO_ENTRY
(BITSELECT, BLISS_BITSELECT_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
```

```
! CONVERT gets called to convert subscripts to integer type.
! It also gets called to convert expressions in FOR loops or
! REPEAT counts, or expressions in WHILE or IF statements, to
! integer type. As far as I know, conversion to type L is the
! only conversion we see for BLISS. The normal hierarchy table
! should thus be adequate.
```

```
OPERATOR_INFO_ENTRY
(CONVERT, TABLEBASE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
```

```
! The fetch operator is the one that has the fetch flag set to TRUE.
```

```
OPERATOR_INFO_ENTRY
(INDIRECT, BLISS_INDIRECT_TABLE, BLISS_HIER_TABLE, TABLEBASE, TRUE),
```

```
! DEPOSIT uses its own hierarchy table, allowing DEPOSIT of any type
! into any type. The fetch flag is still false, indicating that
! no implicit fetch is done on the right-hand-side of the deposit.
! E.g., DEP X = Y will put the address of Y into X; DEP X = .Y
! will put the value of Y into X.
```

```
OPERATOR_INFO_ENTRY
(DEPOSIT, TABLEBASE, BLISS_HIERD_TABLE, TABLEBASE, FALSE),
```

```
! The identity operator is called at the end of an evaluate if
! we still have a primary, e.g., EVAL X will call DBGSEVAL_LANG_OPERATOR
! with "IDENTITY" and we can then do our thing. In the BLISS case,
! "doing our thing" means calling DBGSPRIM_TO_ADDR, and returning
! that descriptor.
```

```
! The reason for an "identity" operator is to ensure that EVAL A
! will go through the same code paths as, say, EVAL A+0 or EVAL +A
```

```
! The identity operator uses no tables.
```

```
OPERATOR_INFO_ENTRY
(IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE, FALSE)
);
```

## C OPERATOR INFORMATION TABLES

This section contains the Operator Routine and Type tables needed to evaluate expressions in the C language.

## C constants:

Integer: decimal, octal, hexadecimal. Double, String

## C Data Types:

- Integer (signed, unsigned)
- char ( 8-bit byte), short (16-bit integer), int (32-bit integer)
- Floating-point numbers
- float (F\_float), double (D\_float)
- Enum values
- Scalars of a user-defined type
- Pointers (typed)
- 32-bit addresses of other variables

## C aggregates:

- Array
- Structure
- Union

## Expressions:

- Primary
- Negating Arithmetic (-E, TC applied)  
E: an E of any arithmetic type.  
Result: the arithmetic negative of the expression. (The negative of an unsigned quantity is computed by subtracting its value from  $2^{*32}$ )
- Negating Logical (!E, TC applied)  
E: pointer (or other address-valued E, array), or an E of any arithmetic type. Result: the logical negative of the expression, result type is int.
- Incrementing and Decrementing Variables (--, ++)
- Computing address (&lvalue)  
Result: the address of the object to which the lvalue refers. (The & may not be applied to register or to bit fields in structure or union).
- Dereferencing Pointers (\*E)  
E: Pointer or other address-valued E.  
Result: a reference to the object to which the expression points, the type of the addressed object is the type of the result.
- One's complement (~E, TC applied)  
E: integer or character
- Additive - (+, -, TC applied)  
1. Operand: Address of an array element and a value of any integral type can be added (integer is converted to an address offset by by integer \* length of the addressed object). Result: the address

```
1859 1980 1
1860 1981 1
1861 1982 1
1862 1983 1
1863 1984 1
1864 1985 1
1865 1986 1
1866 1987 1
1867 1988 1
1868 1989 1
1869 1990 1
1870 1991 1
1871 1992 1
1872 1993 1
1873 1994 1
1874 1995 1
1875 1996 1
1876 1997 1
1877 1998 1
1878 1999 1
1879 2000 1
1880 2001 1
1881 2002 1
1882 2003 1
1883 2004 1
1884 2005 1
1885 2006 1
1886 2007 1
1887 2008 1
1888 2009 1
1889 2010 1
1890 2011 1
1891 2012 1
1892 2013 1
1893 2014 1
1894 2015 1
1895 2016 1
1896 2017 1
1897 2018 1
1898 2019 1
1899 2020 1
1900 2021 1
1901 2022 1
1902 2023 1
1903 2024 1
1904 2025 1
1905 2026 1
1906 2027 1
1907 2028 1
1908 2029 1
1909 2030 1
1910 2031 1
1911 2032 1
1912 2033 1
1913 P 2034 1
1914 2035 1
1915 2036 1
```

of an object of the same type.

2. Operand: A value of any integral type may be subtracted from a pointer or address.

3. Operand: enum + enum, or enum - enum. Result: int

4. If two addresses of objects of the same type are subtracted, the result is int.

. Multiplicative - (\*, /, %: mod, TC applied)  
Operand: integral mod integral.

. Equality - (==, !=, TC applied) Result: type int.  
1. Two pointers or addresses (if they identify the same storage location then they are equal).  
2. A pointer or address can be compared with an integer.

. Relational - (<, >, <=, >=, TC applied) Result: type int.  
same as above.

. Bitwise - (&: and, ^: xor, |:or, TC applied)  
Operand: both must be integrals.

. Logical - (&&: and, ||: or) Result: type int.  
Operand: fundamental types or a pointer, or address-valued E.

. Shift - (<<, >>, TC applied)  
Operand: both must be integral. the right-hand operand --> int, and the type of the result is the type of the left operand.  
E1 << E2: the value of E1 shifted to the left by E2 bits, vacated bits are cleared.  
E1 >> E2: the value of E1 shifted to the right by E2 bits. Vacated bits are cleared if E1 is unsigned, else, vacated bits are filled with a copy of E1's sign bit.  
(The result of the shift is undefined if E2 is negative, or the value of E2 > 32 bits)

. Assignment, conditional, and comma (we do not support)

Type Conversion: (operands of different types appear in an expression)

. Arithmetic  
1. char or short (signed or unsigned) --> int (signed, or unsigned)  
if float --> d float. Char is treated as signed.  
2. If either operand is double, the other --> double. Result: double.  
3. If either operand is unsigned, the other --> unsigned. Result: unsigned.  
4. Otherwise, both operands must be int. Result: integer.  
5. Whenever an unsigned integer and a signed integer are combined, signed --> signed int --> unsigned. Result: unsigned.  
6. For some operators require integers as operands, f or d --> int.

! Define the Type Conversion Information Table for C.  
C has no special rules for type conversions, so we do not have a language specific type conversion table.

CONVERSION INFO TABLE (C CVTINFO TABLE,  
CONVERSION\_INFO\_ENTRY (TABLEBASE, TABLEBASE));

```
1916 2037 1
1917 2038 1
1918 2039 1
1919 2040 1
1920 2041 1
1921 2042 1
1922 2043 1
1923 2044 1
1924 2045 1
1925 2046 1
1926 2047 1
1927 2048 1
1928 2049 1
1929 2050 1
1930 2051 1
1931 2052 1
1932 2053 1
1933 2054 1
1934 2055 1
1935 2056 1
1936 2057 1
1937 2058 1
1938 2059 1
1939 2060 1
1940 P 2061 1
1941 P P 2062 1
1942 P P P 2063 1
1943 P P P 2064 1
1944 P P P 2065 1
1945 P P P 2066 1
1946 P P P 2067 1
1947 P P P 2068 1
1948 P P P 2069 1
1949 P P P 2070 1
1950 P P P 2071 1
1951 P P P 2072 1
1952 P P P 2073 1
1953 P P P 2074 1
1954 P P P 2075 1
1955 P P P 2076 1
1956 P P P 2077 1
1957 P P P 2078 1
1958 P P P 2079 1
1959 P P P 2080 1
1960 P P P 2081 1
1961 P P P 2082 1
1962 P P P 2083 1
1963 P P P 2084 1
1964 P P P 2085 1
1965 P P P 2086 1
1966 P P P 2087 1
1967 P P P 2088 1
1968 P P P 2089 1
1969 P P P 2090 1
1970 P P P 2091 1
1971 P P P 2092 1
1972 P P P 2093 1

! Define the Type Hierarchy Table for C.
! This table defines what is referred to as the 'usual type conversion rules'
! in the C manual. These rules state, basically, that:
!
! char -> longword integer
! short integer -> longword integer (with the same sign attribute)
! signed integer -> unsigned
! integer -> float
! float -> double float
!
! The first edge, T->B, is there so that variables declared as CHAR can
! be treated as integers, as C allows. The conversion should check that
! the length of the char string is 1.
!
! We also include an edge for ENUM->L. This will allow any arithmetic with
! enumeration types that is also allowed for integers. This may be a more
! permissive implementation than the language allows.
!
! We also include V -> LU, VU -> LU, SV -> L, and SVU -> L.
! C declares components of a packed record to be of type V, VU, SV, or SVU,
! but operations on these are just integer operations. So we convert
! these to integer.
!
TYPE_HIERARCHY_TABLE (C_HIER_TABLE,
  TYPE_GRAPH_EDGE (T, B),
  TYPE_GRAPH_EDGE (ENUM, L),
  TYPE_GRAPH_EDGE (V, LU),
  TYPE_GRAPH_EDGE (VU, LU),
  TYPE_GRAPH_EDGE (SV, L),
  TYPE_GRAPH_EDGE (SVU, L),
  TYPE_GRAPH_EDGE (B, W),
  TYPE_GRAPH_EDGE (W, L),
  TYPE_GRAPH_EDGE (BU, WU),
  TYPE_GRAPH_EDGE (WU, LU),
  TYPE_GRAPH_EDGE (L, LU),
  TYPE_GRAPH_EDGE (L, D),
  TYPE_GRAPH_EDGE (LU, D),
  TYPE_GRAPH_EDGE (F, D),
  0);

! The HIERD table defines what pairs are legal in a DEPOSIT.
! Give a circular graph which allows DEPOSIT any-any.
!
TYPE_HIERARCHY_TABLE (C_HIERD_TABLE,
  TYPE_GRAPH_EDGE (T, B),
  TYPE_GRAPH_EDGE (B, BU),
  TYPE_GRAPH_EDGE (BU, W),
  TYPE_GRAPH_EDGE (W, WU),
  TYPE_GRAPH_EDGE (WU, LU),
  TYPE_GRAPH_EDGE (LU, L),
  TYPE_GRAPH_EDGE (L, F),
  TYPE_GRAPH_EDGE (F, D),
```

```
1973 P 2094 1 TYPE_GRAPH_EDGE (D, ENUM)
1974 P 2095 1 TYPE_GRAPH_EDGE (ENUM, TPTR),
1975 P 2096 1 TYPE_GRAPH_EDGE (TPTR, V),
1976 P 2097 1 TYPE_GRAPH_EDGE (V, SV),
1977 P 2098 1 TYPE_GRAPH_EDGE (SV, VU),
1978 P 2099 1 TYPE_GRAPH_EDGE (VU, SVU),
1979 P 2100 1 TYPE_GRAPH_EDGE (SVU, T),
1980 2101 1 0);
1981 2102 1
1982 2103 1
1983 2104 1 ! Add, subtract, multiply, divide, unary minus
1984 2105 1 ! These accept all numeric types.
1985 2106 1
1986 2107 1 ! Define the Operator Routine Table for C Addition.
1987 2108 1 ! Add also has special cases for TPTR + L, which is a special kind
1988 2109 1 ! of addition in which the integer is scaled to the size of the
1989 2110 1 ! pointed-to object.
1990 2111 1
1991 P 2112 1 OPERATOR_ROUTINE_TABLE (C_ADD_TABLE,
1992 P 2113 1 OPERATOR_ROUTINE (L, C, L, ADD_L_L),
1993 P 2114 1 OPERATOR_ROUTINE (LU, LU, LU, ADD_LU_LU),
1994 P 2115 1 OPERATOR_ROUTINE (D, D, D, ADD_D_D),
1995 P 2116 1 OPERATOR_ROUTINE (TPTR, LU, TPTR, ADD_TPTR_L),
1996 2117 1 OPERATOR_ROUTINE (LU, TPTR, TPTR, ADD_TPTR_L));
1997 2118 1
1998 2119 1
1999 2120 1 ! Define the Operator Routine Table for C Subtraction.
2000 2121 1 ! There is a special case routine for TPTR - L, which is a special kind
2001 2122 1 ! of subtraction in which the integer is scaled to the size of the
2002 2123 1 ! pointed-to object. Also, TPTR-TPTR is another special case in which
2003 2124 1 ! the result is scaled.
2004 2125 1
2005 P 2126 1 OPERATOR_ROUTINE_TABLE (C_SUB_TABLE,
2006 P 2127 1 OPERATOR_ROUTINE (L, C, L, SUB_L_L),
2007 P 2128 1 OPERATOR_ROUTINE (LU, LU, LU, SUB_LU_LU),
2008 P 2129 1 OPERATOR_ROUTINE (D, D, D, SUB_D_D),
2009 P 2130 1 OPERATOR_ROUTINE (TPTR, LU, TPTR, SUB_TPTR_L),
2010 2131 1 OPERATOR_ROUTINE (TPTR, TPTR, L, SUB_TPTR_TPTR));
2011 2132 1
2012 2133 1
2013 2134 1 ! Define the Operator Routine Table for C Multiplication.
2014 2135 1
2015 P 2136 1 OPERATOR_ROUTINE_TABLE (C_MUL_TABLE,
2016 P 2137 1 OPERATOR_ROUTINE (L, C, L, MUL_L_L),
2017 P 2138 1 OPERATOR_ROUTINE (LU, LU, LU, MUL_LU_LU),
2018 2139 1 OPERATOR_ROUTINE (D, D, D, MUL_D_D));
2019 2140 1
2020 2141 1
2021 2142 1 ! Define the Operator Routine Table for C Division.
2022 2143 1
2023 P 2144 1 OPERATOR_ROUTINE_TABLE (C_DIV_TABLE,
2024 P 2145 1 OPERATOR_ROUTINE (L, C, L, DIV_L_L),
2025 P 2146 1 OPERATOR_ROUTINE (LU, LU, LU, DIV_LU_LU),
2026 2147 1 OPERATOR_ROUTINE (D, D, D, DIV_D_D));
2027 2148 1
2028 2149 1
2029 2150 1 ! Define the Operator Routine Table for C Unary Minus (Negating Arithmetic).
```

```
2030 2151 1 !
2031 P 2152 1 OPERATOR_ROUTINE_TABLE (C_UNARY_MINUS_TABLE,
2032 P 2153 1
2033 P 2154 1 ! The following are not language dependent types. This is needed for DEBUG
2034 P 2155 1 ! types. For example, DEP/QUAD L= +1.
2035 P 2156 1
2036 P 2157 1 OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
2037 P 2158 1 OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
2038 P 2159 1 OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
2039 P 2160 1 OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
2040 P 2161 1 OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
2041 P 2162 1 OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
2042 P 2163 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
2043 P 2164 1 OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
2044 P 2165 1
2045 P 2166 1 OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
2046 P 2167 1 OPERATOR_ROUTINE (LU, LU, LU, UNARY_MINUS_LU),
2047 P 2168 1 OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D));
2048 P 2169 1
2049 P 2170 1
2050 P 2171 1 ! This table is needed for the +/- constant.
2051 P 2172 1
2052 P 2173 1 OPERATOR_ROUTINE_TABLE (C_UNARY_PLUS_TABLE,
2053 P 2174 1
2054 P 2175 1 ! The following are not language dependent types. This is needed for DEBUG
2055 P 2176 1 ! types. For example, DEP/QUAD L= +1.
2056 P 2177 1
2057 P 2178 1 OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
2058 P 2179 1 OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
2059 P 2180 1 OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
2060 P 2181 1 OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
2061 P 2182 1 OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
2062 P 2183 1 OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
2063 P 2184 1 OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
2064 P 2185 1 OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
2065 P 2186 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
2066 P 2187 1 OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O));
2067 P 2188 1
2068 P 2189 1
2069 P 2190 1 ! Define the Operator Routine Table for C Modulus.
2070 P 2191 1 ! Modulus is only defined for integer types.
2071 P 2192 1 ! Note: The C Modulus function is really the remainder function.
2072 P 2193 1
2073 P 2194 1 OPERATOR_ROUTINE_TABLE (C_MOD_TABLE,
2074 P 2195 1 OPERATOR_ROUTINE (L, L, L, REM_L_L),
2075 P 2196 1 OPERATOR_ROUTINE (LU, LU, LU, REM_LU_LU));
2076 P 2197 1
2077 P 2198 1
2078 P 2199 1 ! Relational operators. These accept all three numeric types,
2079 P 2200 1 ! and also pointer type (which is just treated as integer).
2080 P 2201 1
2081 P 2202 1 ! Define the Operator Routine Table for C Equal.
2082 P 2203 1 ! Signed and unsigned equal are the same so they both use the EQL_L_L
2083 P 2204 1 ! routine.
2084 P 2205 1
2085 P 2206 1 OPERATOR_ROUTINE_TABLE (C_EQL_TABLE,
2086 P 2207 1 OPERATOR_ROUTINE (TPTR, LD, L, EQL_L_L),
```

```
2087      P 2208 1 OPERATOR_ROUTINE (LU, TPTR, L, EQL_L_L),
2088      P 2209 1 OPERATOR_ROUTINE (TPTR, TPTR, L, EQL_L_L),
2089      P 2210 1 OPERATOR_ROUTINE (L, L, L, EQL_L_L),
2090      P 2211 1 OPERATOR_ROUTINE (LU, LU, L, EQL_L_L),
2091      P 2212 1 OPERATOR_ROUTINE (D, D, L, EQL_D_DT);
2092      P 2213 1
2093      P 2214 1
2094      P 2215 1 :: Define the Operator Routine Table for C Not Equal.
2095      P 2216 1 Signed and unsigned not-equal are the same so they both use the EQL_L_L
2096      P 2217 1 routine.
2097      P 2218 1
2098      P 2219 1 OPERATOR_ROUTINE_TABLE (C NEQ_TABLE,
2099      P 2220 1 OPERATOR_ROUTINE (TPTR, LU, L, NEQ_L_L),
2100      P 2221 1 OPERATOR_ROUTINE (LU, TPTR, L, NEQ_L_L),
2101      P 2222 1 OPERATOR_ROUTINE (TPTR, TPTR, L, NEQ_L_L),
2102      P 2223 1 OPERATOR_ROUTINE (L, L, L, NEQ_L_L),
2103      P 2224 1 OPERATOR_ROUTINE (LU, LU, L, NEQ_L_L),
2104      P 2225 1 OPERATOR_ROUTINE (D, D, L, NEQ_D_DT);
2105      P 2226 1
2106      P 2227 1
2107      P 2228 1 :: Define the Operator Routine Table for C Less Than.
2108      P 2229 1 Unsigned less than used a different routine than signed less than.
2109      P 2230 1
2110      P 2231 1 OPERATOR_ROUTINE_TABLE (C LSS_TABLE,
2111      P 2232 1 OPERATOR_ROUTINE (TPTR, L, L, LSS_L_L),
2112      P 2233 1 OPERATOR_ROUTINE (TPTR, LU, L, LSS_CU_LU),
2113      P 2234 1 OPERATOR_ROUTINE (L, TPTR, L, LSS_C_LT),
2114      P 2235 1 OPERATOR_ROUTINE (LU, TPTR, L, LSS_CU_LU),
2115      P 2236 1 OPERATOR_ROUTINE (TPTR, TPTR, L, LSS_C_L),
2116      P 2237 1 OPERATOR_ROUTINE (L, L, L, LSS_L_L),
2117      P 2238 1 OPERATOR_ROUTINE (LU, LU, L, LSS_LU_LU),
2118      P 2239 1 OPERATOR_ROUTINE (D, D, L, LSS_D_DT);
2119      P 2240 1
2120      P 2241 1
2121      P 2242 1 :: Define the Operator Routine Table for C Greater Than.
2122      P 2243 1 Unsigned greater than uses a different routine than signed greater than.
2123      P 2244 1
2124      P 2245 1 OPERATOR_ROUTINE_TABLE (C GTR_TABLE,
2125      P 2246 1 OPERATOR_ROUTINE (TPTR, L, L, GTR_L_L),
2126      P 2247 1 OPERATOR_ROUTINE (TPTR, LU, L, GTR_CU_LU),
2127      P 2248 1 OPERATOR_ROUTINE (L, TPTR, L, GTR_C_LT),
2128      P 2249 1 OPERATOR_ROUTINE (LU, TPTR, L, GTR_CU_LU),
2129      P 2250 1 OPERATOR_ROUTINE (TPTR, TPTR, L, GTR_C_L),
2130      P 2251 1 OPERATOR_ROUTINE (L, L, L, GTR_L_L),
2131      P 2252 1 OPERATOR_ROUTINE (LU, LU, L, GTR_LU_LU),
2132      P 2253 1 OPERATOR_ROUTINE (D, D, L, GTR_D_DT);
2133      P 2254 1
2134      P 2255 1
2135      P 2256 1 :: Define the Operator Routine Table for C Less Than or Equal To.
2136      P 2257 1 Unsigned less than/equal to uses a different routine than signed
2137      P 2258 1 less than/equal to.
2138      P 2259 1
2139      P 2260 1 OPERATOR_ROUTINE_TABLE (C LEQ_TABLE,
2140      P 2261 1 OPERATOR_ROUTINE (TPTR, L, L, LEQ_L_L),
2141      P 2262 1 OPERATOR_ROUTINE (TPTR, LU, L, LEQ_CU_LU),
2142      P 2263 1 OPERATOR_ROUTINE (L, TPTR, L, LEQ_C_LT),
2143      P 2264 1 OPERATOR_ROUTINE (LU, TPTR, L, LEQ_CU_LU).
```

```
2144 P 2265 1 OPERATOR_ROUTINE (TPTR, TPTR, L, LEQ_L_L),
2145 PP 2266 1 OPERATOR_ROUTINE (L, L, L, LEQ_L_L),
2146 P 2267 1 OPERATOR_ROUTINE (LU, LU, L, LEQ_LU_LU),
2147 2268 1 OPERATOR_ROUTINE (D, D, L, LEQ_D_D);
2148 2269 1
2149 2270 1
2150 2271 1 ! Define the Operator Routine Table for C Greater Than or Equal.
2151 2272 1 ! Unsigned greater than or equal to uses a different routine than
2152 2273 1 ! signed greater than or equal to.
2153 2274 1
2154 2275 1 OPERATOR_ROUTINE_TABLE (C_GEQ_TABLE,
2155 2276 1 OPERATOR_ROUTINE (TPTR, L, L, GEQ_L_L),
2156 2277 1 OPERATOR_ROUTINE (TPTR, LU, L, GEQ_LU_LU),
2157 2278 1 OPERATOR_ROUTINE (L, TPTR, L, GEQ_L_L),
2158 2279 1 OPERATOR_ROUTINE (LU, TPTR, L, GEQ_LU_LU),
2159 2280 1 OPERATOR_ROUTINE (TPTR, TPTR, L, GEQ_L_L),
2160 2281 1 OPERATOR_ROUTINE (L, L, L, GEQ_L_L),
2161 2282 1 OPERATOR_ROUTINE (LU, LU, L, GEQ_LU_LU),
2162 2283 1 OPERATOR_ROUTINE (D, D, L, GEQ_D_D);
2163 2284 1
2164 2285 1
2165 2286 1 ! Bitwise operators.
2166 2287 1 ! These accept only integer data types (no float).
2167 2288 1
2168 2289 1 ! Define the Operator Routine Table for C Bitwise And.
2169 2290 1
2170 P 2291 1 OPERATOR_ROUTINE_TABLE (C_BIT_AND_TABLE,
2171 P 2292 1 OPERATOR_ROUTINE (L, L, L, BIT_AND_L_L),
2172 2293 1 OPERATOR_ROUTINE (LU, LU, LU, BIT_AND_L_L));
2173 2294 1
2174 2295 1
2175 2296 1 ! Define the Operator Routine Table for C Bitwise Or.
2176 2297 1
2177 P 2298 1 OPERATOR_ROUTINE_TABLE (C_BIT_OR_TABLE,
2178 P 2299 1 OPERATOR_ROUTINE (L, L, L, BIT_OR_L_L),
2179 2300 1 OPERATOR_ROUTINE (LU, LU, LU, BIT_OR_L_L));
2180 2301 1
2181 2302 1
2182 2303 1 ! Define the Operator Routine Table for C Bitwise Xor.
2183 2304 1
2184 P 2305 1 OPERATOR_ROUTINE_TABLE (C_BIT_XOR_TABLE,
2185 P 2306 1 OPERATOR_ROUTINE (L, L, L, BIT_XOR_L_L),
2186 2307 1 OPERATOR_ROUTINE (LU, LU, LU, BIT_XOR_L_L));
2187 2308 1
2188 2309 1 ! Define the Operator Routine Table for C Bitwise Not (One's Complement).
2189 2310 1
2190 P 2311 1 OPERATOR_ROUTINE_TABLE (C_BIT_NOT_TABLE,
2191 P 2312 1 OPERATOR_ROUTINE (L, L, L, BIT_NOT_L),
2192 2313 1 OPERATOR_ROUTINE (LU, LU, LU, BIT_NOT_L));
2193 2314 1
2194 2315 1
2195 2316 1 ! Logical operations.
2196 2317 1 ! These just do the Boolean operations with TRUE <=> not zero,
2197 2318 1 ! FALSE <=> zero.
2198 2319 1 ! For mixed int-float logical operations, we convert both to float.
2199 2320 1 ! This is not identical to what the compiler does. But hopefully,
2200 2321 1 ! conversion to float should preserve the zero/notzero characteristic.
```

```
2201 2322 1 1 so we should get the same final answer except in obscure boundary
2202 2323 1 1 conditions.
2203 2324 1 1 We also allow pointers, which are just treated as integer.
2204 2325 1 1
2205 2326 1 1 Define the Operator Routine Table for C Logical And.
2206 2327 1 1
2207 2328 1 1 OPERATOR ROUTINE TABLE (C AND TABLE,
2208 2329 1 1 OPERATOR_ROUTINE (TPTR, TPTR, L, AND_L_L),
2209 2330 1 1 OPERATOR_ROUTINE (TPTR, LU, L, AND_L_L),
2210 2331 1 1 OPERATOR_ROUTINE (LU, TPTR, L, AND_L_L),
2211 2332 1 1 OPERATOR_ROUTINE (L, L, L, AND_L_L),
2212 2333 1 1 OPERATOR_ROUTINE (LU, LU, L, AND_L_L),
2213 2334 1 1 OPERATOR_ROUTINE (D, D, L, AND_D_DT);
2214 2335 1 1
2215 2336 1 1
2216 2337 1 1 Define the Operator Routine Table for C Logical Or.
2217 2338 1 1
2218 2339 1 1 OPERATOR ROUTINE TABLE (C OR TABLE,
2219 2340 1 1 OPERATOR_ROUTINE (TPTR, TPTR, L, OR_L_L),
2220 2341 1 1 OPERATOR_ROUTINE (TPTR, LU, L, OR_L_L),
2221 2342 1 1 OPERATOR_ROUTINE (LU, TPTR, L, OR_L_L),
2222 2343 1 1 OPERATOR_ROUTINE (L, L, L, OR_L_L),
2223 2344 1 1 OPERATOR_ROUTINE (LU, LU, L, OR_L_L),
2224 2345 1 1 OPERATOR_ROUTINE (D, D, L, OR_D_DT);
2225 2346 1 1
2226 2347 1 1
2227 2348 1 1 Define the Operator Routine Table for C Logical Not.
2228 2349 1 1
2229 2350 1 1 OPERATOR ROUTINE TABLE (C NOT TABLE,
2230 2351 1 1 OPERATOR_ROUTINE (TPTR, TPTR, L, NOT_L),
2231 2352 1 1 OPERATOR_ROUTINE (L, L, L, NOT_L),
2232 2353 1 1 OPERATOR_ROUTINE (LU, LU, L, NOT_L),
2233 2354 1 1 OPERATOR_ROUTINE (D, D, L, NOT_DT);
2234 2355 1 1
2235 2356 1 1
2236 2357 1 1 Shift operators.
2237 2358 1 1 These accept only integer types.
2238 2359 1 1
2239 2360 1 1 Define the Operator Routine Table for C Left Shift.
2240 2361 1 1
2241 2362 1 1 OPERATOR ROUTINE TABLE (C SHIFT LEFT TABLE,
2242 2363 1 1 OPERATOR_ROUTINE (L, L, L, SHIFT_LEFT_L_L),
2243 2364 1 1 OPERATOR_ROUTINE (LU, LU, LU, SHIFT_LEFT_L_L));
2244 2365 1 1
2245 2366 1 1
2246 2367 1 1 Define the Operator Routine Table for C Right Shift.
2247 2368 1 1 Unsigned right shift is different from signed. For unsigned right
2248 2369 1 1 shift, we always shift in zeros. For signed right shift, we shift
2249 2370 1 1 in copies of the sign bit.
2250 2371 1 1
2251 2372 1 1 OPERATOR ROUTINE TABLE (C SHIFT RT TABLE,
2252 2373 1 1 OPERATOR_ROUTINE (L, L, L, SHIFT_RT_L_L),
2253 2374 1 1 OPERATOR_ROUTINE (LU, LU, LU, SHIFT_RT_LU_LU));
2254 2375 1 1
2255 2376 1 1
2256 2377 1 1 Define Operator Routine Tables for ++X X++ --X X--
2257 2378 1 1
```

```
2258 2379 1 OPERATOR ROUTINE TABLE (C PRE_INCR_TABLE,
2259 2380 1 OPERATOR_ROUTINE (L, C, L, PRE_INCR_L),
2260 2381 1 OPERATOR_ROUTINE (LU, LU, LU, PRE_INCR_LU),
2261 2382 1 OPERATOR_ROUTINE (D, D, D, PRE_INCR_D),
2262 2383 1 OPERATOR_ROUTINE (TPTR, TPTR, TPTR, PRE_INCR_TPTR));
2263 2384
2264 2385 1 OPERATOR ROUTINE TABLE (C POST_INCR_TABLE,
2265 2386 1 OPERATOR_ROUTINE (L, C, L, POST_INCR_L),
2266 2387 1 OPERATOR_ROUTINE (LU, LU, LU, POST_INCR_LU),
2267 2388 1 OPERATOR_ROUTINE (D, D, D, POST_INCR_D),
2268 2389 1 OPERATOR_ROUTINE (TPTR, TPTR, TPTR, POST_INCR_TPTR));
2269 2390
2270 2391 1 OPERATOR ROUTINE TABLE (C PRE_DECR_TABLE,
2271 2392 1 OPERATOR_ROUTINE (L, C, L, PRE_DECR_L),
2272 2393 1 OPERATOR_ROUTINE (LU, LU, LU, PRE_DECR_LU),
2273 2394 1 OPERATOR_ROUTINE (D, D, D, PRE_DECR_D),
2274 2395 1 OPERATOR_ROUTINE (TPTR, TPTR, TPTR, PRE_DECR_TPTR));
2275 2396
2276 2397 1 OPERATOR ROUTINE TABLE (C POST_DECR_TABLE,
2277 2398 1 OPERATOR_ROUTINE (L, C, L, POST_DECR_L),
2278 2399 1 OPERATOR_ROUTINE (LU, LU, LU, POST_DECR_LU),
2279 2400 1 OPERATOR_ROUTINE (D, D, D, POST_DECR_D),
2280 2401 1 OPERATOR_ROUTINE (TPTR, TPTR, TPTR, POST_DECR_TPTR));
2281 2402
2282 2403
2283 2404 1 Define the Operator Routine Table for C Address Of.
2284 2405 1 The address-of operator will have the FETCH_FLAG off in the operator
2285 2406 1 information table. This means that PRIM_TO_ADDR will get called.
2286 2407 1 The value of the result will thus be the address of the operand.
2287 2408 1 The type of the result is "pointer to xxx", where "xxx" is the type
2288 2409 1 of the operand. This means that the the routine which is called to
2289 2410 1 do "address_of" must construct a typeid for the result.
2290 2411
2291 2412 1 OPERATOR ROUTINE TABLE (C ADDRESS_TABLE,
2292 2413 1 OPERATOR_ROUTINE (L, C, TPTR, ADDRESS_L));
2293 2414
2294 2415
2295 2416 1 Define the Operator Routine Table for the size of operator.
2296 2417 1 Size-of will look up the typeid in the symbol table and determine
2297 2418 1 the declared size of the object. The FETCH_FLAG will be FALSE, so
2298 2419 1 PRIM_TO_ADDR will be called instead of PRIM_TO_VAL. This is
2299 2420 1 because we do not need the value of the object.
2300 2421
2301 2422 1 OPERATOR ROUTINE TABLE (C SIZEOF_TABLE,
2302 2423 1 OPERATOR_ROUTINE (L, C, L, SIZEOF_L));
2303 2424
2304 2425
2305 2426 1 Define the Operator Routine Table for C Indirection (Dereferencing Pointers).
2306 2427 1 The indirection operator (*) will do a pointer dereference of
2307 2428 1 its argument. The type of the returned object depends on the
2308 2429 1 type of the pointer. (We will have to do a symbol table lookup
2309 2430 1 on the TYPEID).
2310 2431
2311 2432 1 OPERATOR ROUTINE TABLE (C INDIRECT_TABLE,
2312 2433 1 OPERATOR_ROUTINE (TPTR, TPTR, UNKNOWN, INDIRECT_TPTR));
2313 2434
2314 2435
```

```
2315 2436 1 ! Define the Operator Information Table for C.
2316 2437 1
2317 2438 1 OPERATOR_INFO_TABLE (C_OPINFO_TABLE,
2318 2439 1
2319 2440 1 ! All C operators use the same hierarchy table, and have no
2320 2441 1 ! type incompatibility table.
2321 2442 1
2322 2443 1 ! Arithmetic operators.
2323 2444 1
2324 2445 1 OPERATOR_INFO_ENTRY
2325 2446 1 (ADD, C_ADD_TABLE, C_HIER_TABLE, TABLEBASE),
2326 2447 1 OPERATOR_INFO_ENTRY
2327 2448 1 (SUBTRACT, C_SUB_TABLE, C_HIER_TABLE, TABLEBASE),
2328 2449 1 OPERATOR_INFO_ENTRY
2329 2450 1 (MULTIPLY, C_MUL_TABLE, C_HIER_TABLE, TABLEBASE),
2330 2451 1 OPERATOR_INFO_ENTRY
2331 2452 1 (DIVIDE, C_DIV_TABLE, C_HIER_TABLE, TABLEBASE),
2332 2453 1 OPERATOR_INFO_ENTRY
2333 2454 1 (REMAINDER, C_MOD_TABLE, C_HIER_TABLE, TABLEBASE),
2334 2455 1 OPERATOR_INFO_ENTRY
2335 2456 1 (UNARY_MINUS, C_UNARY_MINUS_TABLE, C_HIER_TABLE, TABLEBASE),
2336 2457 1
2337 2458 1 ! This is needed for /Qualifier for depositing the +/- constant.
2338 2459 1
2339 2460 1 OPERATOR_INFO_ENTRY
2340 2461 1 (UNARY_PLUS, C_UNARY_PLUS_TABLE, C_HIER_TABLE, TABLEBASE),
2341 2462 1
2342 2463 1 ! Relational operators.
2343 2464 1
2344 2465 1 OPERATOR_INFO_ENTRY
2345 2466 1 (EQUAL, C_EQL_TABLE, C_HIER_TABLE, TABLEBASE),
2346 2467 1 OPERATOR_INFO_ENTRY
2347 2468 1 (NOT_EQUAL, C_NEQ_TABLE, C_HIER_TABLE, TABLEBASE),
2348 2469 1 OPERATOR_INFO_ENTRY
2349 2470 1 (LESS_THAN, C_LSS_TABLE, C_HIER_TABLE, TABLEBASE),
2350 2471 1 OPERATOR_INFO_ENTRY
2351 2472 1 (GTR_THAN, C_GTR_TABLE, C_HIER_TABLE, TABLEBASE),
2352 2473 1 OPERATOR_INFO_ENTRY
2353 2474 1 (LESS_EQUAL, C_LEQ_TABLE, C_HIER_TABLE, TABLEBASE),
2354 2475 1 OPERATOR_INFO_ENTRY
2355 2476 1 (GTR_EQUAL, C_GEQ_TABLE, C_HIER_TABLE, TABLEBASE),
2356 2477 1
2357 2478 1 ! Bitwise logical operators.
2358 2479 1
2359 2480 1 OPERATOR_INFO_ENTRY
2360 2481 1 (BIT_AND, C_BIT_AND_TABLE, C_HIER_TABLE, TABLEBASE),
2361 2482 1 OPERATOR_INFO_ENTRY
2362 2483 1 (BIT_OR, C_BIT_OR_TABLE, C_HIER_TABLE, TABLEBASE),
2363 2484 1 OPERATOR_INFO_ENTRY
2364 2485 1 (BIT_XOR, C_BIT_XOR_TABLE, C_HIER_TABLE, TABLEBASE),
2365 2486 1 OPERATOR_INFO_ENTRY
2366 2487 1 (BIT_NOT, C_BIT_NOT_TABLE, C_HIER_TABLE, TABLEBASE),
2367 2488 1
2368 2489 1 ! Logical operators.
2369 2490 1
2370 2491 1 OPERATOR_INFO_ENTRY
2371 2492 1 (SHORT_AND, C_AND_TABLE, C_HIER_TABLE, TABLEBASE),
```

```
2372 P 2493 1 OPERATOR_INFO_ENTRY
2373 P 2494 1 (SHORT_OR, C_OR_TABLE, C_HIER_TABLE, TABLEBASE),
2374 P 2495 1 OPERATOR_INFO_ENTRY
2375 P 2496 1 (NOT, C_NOT_TABLE, C_HIER_TABLE, TABLEBASE),
2376 P 2497 1
2377 P 2498 1 ! Shift operators.
2378 P 2499 1
2379 P 2500 1 OPERATOR_INFO_ENTRY
2380 P 2501 1 (LEFT_SHIFT, C_SHIFT_LEFT_TABLE, C_HIER_TABLE, TABLEBASE),
2381 P 2502 1 OPERATOR_INFO_ENTRY
2382 P 2503 1 (RIGHT_SHIFT, C_SHIFT_RT_TABLE, C_HIER_TABLE, TABLEBASE),
2383 P 2504 1
2384 P 2505 1 ! Operators with side effects.
2385 P 2506 1
2386 P 2507 1 OPERATOR_INFO_ENTRY
2387 P 2508 1 (PRE_INCR, C_PRE_INCR_TABLE, C_HIER_TABLE, TABLEBASE),
2388 P 2509 1 OPERATOR_INFO_ENTRY
2389 P 2510 1 (POST_INCR, C_POST_INCR_TABLE, C_HIER_TABLE, TABLEBASE),
2390 P 2511 1 OPERATOR_INFO_ENTRY
2391 P 2512 1 (PRE_DECR, C_PRE_DECR_TABLE, C_HIER_TABLE, TABLEBASE),
2392 P 2513 1 OPERATOR_INFO_ENTRY
2393 P 2514 1 (POST_DECR, C_POST_DECR_TABLE, C_HIER_TABLE, TABLEBASE),
2394 P 2515 1
2395 P 2516 1 ! Operators that work on the SYMID or TYPEID.
2396 P 2517 1
2397 P 2518 1 OPERATOR_INFO_ENTRY
2398 P 2519 1 (ADDRESS_OF, C_ADDRESS_TABLE, C_HIER_TABLE, TABLEBASE, FALSE),
2399 P 2520 1 OPERATOR_INFO_ENTRY
2400 P 2521 1 (SIZEOF, C_SIZEOF_TABLE, C_HIER_TABLE, TABLEBASE, FALSE),
2401 P 2522 1
2402 P 2523 1 ! Indirection
2403 P 2524 1
2404 P 2525 1 OPERATOR_INFO_ENTRY
2405 P 2526 1 (INDIRECT, C_INDIRECT_TABLE, C_HIER_TABLE, TABLEBASE),
2406 P 2527 1
2407 P 2528 1 ! DEPOSIT is used to implement the DEPOSIT command.
2408 P 2529 1
2409 P 2530 1 OPERATOR_INFO_ENTRY
2410 P 2531 1 (DEPOSIT, TABLEBASE, C_HIERD_TABLE, TABLEBASE),
2411 P 2532 1
2412 P 2533 1 ! CONVERT is used for things like converting subscripts.
2413 P 2534 1
2414 P 2535 1 OPERATOR_INFO_ENTRY
2415 P 2536 1 (CONVERT, TABLEBASE, C_HIER_TABLE, TABLEBASE),
2416 P 2537 1
2417 P 2538 1 ! Identity is called at the end of an EVALUATE if we still have
2418 P 2539 1 a Primary.
2419 P 2540 1
2420 P 2541 1 OPERATOR_INFO_ENTRY
2421 P 2542 1 (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE));
2422 P 2543 1
```

## COBOL OPERATOR INFORMATION TABLES

This section contains the Operator Routine and Type tables needed to evaluate expressions in the COBOL language.

The following summarizes the information in the COBOL manual, and from our old cobol support about data types, type conversions and operators. There is further documentation within the tables below, describing exactly how we translate this into the DEBUG tables.

### Cobol Data Types:

.Elementary:	
Alphabetic, Alphanumeric:	(Text string)
Numeric:	
Binary	
PIC 9	-- PIC 9(4) COMP (Scaled W, WU)
PIC 9(5)	-- PIC 9(9) COMP (Scaled L, LU)
PIC 9(10)	-- PIC 9(18) COMP (Scaled Q, QU)
INDEX	(L)
Floating	
COMP-1	(Floating)
COMP-2	(Double)
Packed-Decimal	
COMP-3	(Packed with/without Scaled)
Decimal	
PIC 99...	(Scaled NU)
PIC \$9... LEADING SEPARATE	(Scaled NL)
PIC \$9... LEADING	(Scaled NLO)
PIC \$9... TRAILING SEPARATE	(Scaled NR)
PIC \$9... TRAILING	(Scaled NRO)
.Group	
Alphanumeric	(Text String)
.Edited data	(Text String)

### Expressions:

.Arithmetic	
operator: + - * / (** - not support) unary+ unary-	
operand: numeric literal or numeric elementary identifier	
.Conditional (TRUE, FALSE)	
operator: =, NOT =, >, NOT >, <, NOT <	
operand: elementary identifier, literal, or AE	
.Complex Conditional (TRUE, FALSE)	
operator: NOT, AND, OR	
operand: conditions	

Note: abbreviated forms are not supported.

### Type Conversion:

```
2481 2601 1 SD WU==W -----+
2482 2602 1 SD LU==L -----+
2483 2603 1 SD QU==Q -----+
2484 2604 1 SD NU, NL, NLO, NR, NRO--> P
2485 2605 1 SD F-----+
2486 2606 1 SD D-----+
2487 2607 1
2488 2608 1
2489 2609 1 1. All the integer and float constants are picked up as packed decimal if
2490 2610 1 one of the operand is SD descriptor type.
2491 2611 1
2492 2612 1 2. If one of the operand is Scaled Descriptor or Packed decimal
2493 2613 1 data type the operation is always done in Packed Form. The
2494 2614 1 operation even includes the deposit.
2495 2615 1
2496 2616 1 So in COBOL, one will have the following case:
2497 2617 1 DEP SC2V2=F
2498 2618 1 F converts to P, make a packed decimal place holder for SC2V2
2499 2619 1 (which is word scaled -2), convert P to this place holder,
2500 2620 1 final conversion will convert this place holder to SC2V2 data type.
2501 2621 1
2502 2622 1 EV F+F
2503 2623 1 the operation is done by adding F to F.
2504 2624 1
2505 2625 1 EV F+P
2506 2626 1 the operation is done by adding P to P.
2507 2627 1
2508 2628 1 DEP P=123.123
2509 2629 1 in here 123.123 is picked up as packed decimal constant instead of
2510 2630 1 float.
2511 2631 1 DEP (SD, WU) = -123
2512 2632 1 the value we got is the absolute value.
2513 2633 1
2514 2634 1 EV 1.1+1
2515 2635 1 the operation is done in Packed form.
2516 2636 1
2517 2637 1 Define the COBOL Specific Type Conversion Table. This allows proper
2518 2638 1 handling of COBOL Edited Data type.
2519 2639 1
2520 2640 1 P LANG_CVT_TABLE (COBOL_CVT_TABLE,
2521 2641 1 P LANG_CVT_ENTRY (COB_PICT, PICT, ANY),
2522 2642 1 0);
2523 2643 1
2524 2644 1
2525 2645 1 Define the Type Conversion Information Table for COBOL. No rounding
2526 2646 1 takes place in COBOL.
2527 2647 1
2528 2648 1 P CONVERSION_INFO_TABLE (COBOL_CVTINFO_TABLE,
2529 2649 1 CONVERSION_INFO_ENTRY (TABLEBASE, COBOL_CVT_TABLE));
2530 2650 1
2531 2651 1 Define Type Hierarchy Table for COBOL.
2532 2652 1
2533 2653 1 P TYPE_HIERARCHY_TABLE (COBOL_HIER_TABLE,
2534 2654 1 P TYPE_GRAPH_EDGE (WU, P),
2535 2655 1 P TYPE_GRAPH_EDGE (W, P),
2536 2656 1 P TYPE_GRAPH_EDGE (LU, P),
2537 2657 1 P TYPE_GRAPH_EDGE (L, P),
```

```
2538      TYPE_GRAPH_EDGE (QU, P),
2539      TYPE_GRAPH_EDGE (Q, P),
2540      TYPE_GRAPH_EDGE (NU, P),
2541      TYPE_GRAPH_EDGE (NL, P),
2542      TYPE_GRAPH_EDGE (NLO, P),
2543      TYPE_GRAPH_EDGE (NR, P),
2544      TYPE_GRAPH_EDGE (NRO, P),
2545      TYPE_GRAPH_EDGE (F, D),
2546      TYPE_GRAPH_EDGE (D, P),
2547      0);
2548
2549      ! Define the Type Hierarchy Table for COBOL deposit.
2550
2551      TYPE HIERARCHY_TABLE (COBOL_HIERD_TABLE,
2552      TYPE_GRAPH_EDGE (PICT, QU),
2553      TYPE_GRAPH_EDGE (WU, W),
2554      TYPE_GRAPH_EDGE (W, LU),
2555      TYPE_GRAPH_EDGE (LU, L),
2556      TYPE_GRAPH_EDGE (L, QU),
2557      TYPE_GRAPH_EDGE (QU, Q),
2558      TYPE_GRAPH_EDGE (Q, NU),
2559      TYPE_GRAPH_EDGE (NU, NL),
2560      TYPE_GRAPH_EDGE (NL, NLO),
2561      TYPE_GRAPH_EDGE (NLO, NR),
2562      TYPE_GRAPH_EDGE (NR, NRO),
2563      TYPE_GRAPH_EDGE (NRO, P),
2564      TYPE_GRAPH_EDGE (P, F),
2565      TYPE_GRAPH_EDGE (F, D),
2566      TYPE_GRAPH_EDGE (D, PICT),
2567      0);
2568
2569      ! Define the Operator Routine Table for COBOL addition.
2570
2571      OPERATOR_ROUTINE_TABLE (COBOL_ADD_TABLE,
2572      OPERATOR_ROUTINE (P, P, P, ADD_P_P),
2573      OPERATOR_ROUTINE (F, F, F, ADD_F_F),
2574      OPERATOR_ROUTINE (D, D, D, ADD_D_D));
2575
2576      ! Define the Operator Routine Table for COBOL subtraction.
2577
2578      OPERATOR_ROUTINE_TABLE (COBOL_SUB_TABLE,
2579      OPERATOR_ROUTINE (P, P, P, SUB_P_P),
2580      OPERATOR_ROUTINE (F, F, F, SUB_F_F),
2581      OPERATOR_ROUTINE (D, D, D, SUB_D_D));
2582
2583      ! Define the Operator Routine Table for COBOL multiplication.
2584
2585      OPERATOR_ROUTINE_TABLE (COBOL_MUL_TABLE,
2586      OPERATOR_ROUTINE (P, P, P, MUL_P_P),
2587      OPERATOR_ROUTINE (F, F, F, MUL_F_F),
2588      OPERATOR_ROUTINE (D, D, D, MUL_D_D));
2589
2590      ! Define the Operator Routine Table for COBOL division.
2591
2592      OPERATOR_ROUTINE_TABLE (COBOL_DIV_TABLE,
2593      OPERATOR_ROUTINE (P, P, P, DIV_P_P),
2594      OPERATOR_ROUTINE (F, F, F, DIV_F_F),
```

```
2595 2715 OPERATOR_ROUTINE (D, D, D, DIV_D_D));
2596 2716
2597 2717 ! Define the Operator Routine Table for COBOL unary plus.
2598 2718
2599 2719 OPERATOR_ROUTINE_TABLE (COBOL_UNARY_PLUS_TABLE,
2600 2720
2601 2721 ! The following are not language dependent types. This is needed for DEBUG
2602 2722 types. For example, DEP/QUAD L= +1.
2603 2723
2604 2724 OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
2605 2725 OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
2606 2726 OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
2607 2727 OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
2608 2728 OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
2609 2729 OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
2610 2730 OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
2611 2731
2612 2732 OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
2613 2733 OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
2614 2734 OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D));
2615 2735
2616 2736 ! Define the Operator Routine Table for COBOL unary minus.
2617 2737
2618 2738 OPERATOR_ROUTINE_TABLE (COBOL_UNARY_MINUS_TABLE,
2619 2739
2620 2740 ! The following are not language dependent types. This is needed for DEBUG
2621 2741 types. For example, DEP/QUAD L= +1.
2622 2742
2623 2743 OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
2624 2744 OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
2625 2745 OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
2626 2746 OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
2627 2747 OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
2628 2748 OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
2629 2749 OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
2630 2750
2631 2751 OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
2632 2752 OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
2633 2753 OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D));
2634 2754
2635 2755 ! Define the Operator Routine Table for COBOL =.
2636 2756
2637 2757 OPERATOR_ROUTINE_TABLE (COBOL_EQL_TABLE,
2638 2758 OPERATOR_ROUTINE (T, T, TF, EQL_T_T),
2639 2759 OPERATOR_ROUTINE (P, P, TF, EQL_P_P),
2640 2760 OPERATOR_ROUTINE (F, F, TF, EQL_F_F),
2641 2761 OPERATOR_ROUTINE (D, D, TF, EQL_D_D));
2642 2762
2643 2763 ! Define the Operator Routine Table for COBOL NOT =.
2644 2764
2645 2765 OPERATOR_ROUTINE_TABLE (COBOL_NEQ_TABLE,
2646 2766 OPERATOR_ROUTINE (T, T, TF, NEQ_T_T),
2647 2767 OPERATOR_ROUTINE (P, P, TF, NEQ_P_P),
2648 2768 OPERATOR_ROUTINE (F, F, TF, NEQ_F_F),
2649 2769 OPERATOR_ROUTINE (D, D, TF, NEQ_D_D));
2650 2770
2651 2771 ! Define the Operator Routine Table for COBOL >.
```

```
2652 2772 1 !
2653 P 2773 1 OPERATOR ROUTINE TABLE (COBOL_GTR_TABLE,
2654 P 2774 1 OPERATOR_ROUTINE (T, T, TF, GTR_T_T),
2655 P 2775 1 OPERATOR_ROUTINE (P, P, TF, GTR_P_P),
2656 P 2776 1 OPERATOR_ROUTINE (F, F, TF, GTR_F_F),
2657 2777 1 OPERATOR_ROUTINE (D, D, TF, GTR_D_D));
2658 2778 1
2659 2779 1 ! Define the Operator Routine Table for COBOL NOT <.
2660 2780 1
2661 P 2781 1 OPERATOR ROUTINE TABLE (COBOL_GEQ_TABLE,
2662 P 2782 1 OPERATOR_ROUTINE (T, T, TF, GEQ_T_T),
2663 P 2783 1 OPERATOR_ROUTINE (P, P, TF, GEQ_P_P),
2664 P 2784 1 OPERATOR_ROUTINE (F, F, TF, GEQ_F_F),
2665 2785 1 OPERATOR_ROUTINE (D, D, TF, GEQ_D_D));
2666 2786 1
2667 2787 1 ! Define the Operator Routine Table for COBOL <.
2668 2788 1
2669 P 2789 1 OPERATOR ROUTINE TABLE (COBOL_LSS_TABLE,
2670 P 2790 1 OPERATOR_ROUTINE (T, T, TF, LSS_T_T),
2671 P 2791 1 OPERATOR_ROUTINE (P, P, TF, LSS_P_P),
2672 P 2792 1 OPERATOR_ROUTINE (F, F, TF, LSS_F_F),
2673 2793 1 OPERATOR_ROUTINE (D, D, TF, LSS_D_D));
2674 2794 1
2675 2795 1 ! Define the Operator Routine Table for COBOL NOT >.
2676 2796 1
2677 P 2797 1 OPERATOR ROUTINE TABLE (COBOL_LEQ_TABLE,
2678 P 2798 1 OPERATOR_ROUTINE (T, T, TF, LEQ_T_T),
2679 P 2799 1 OPERATOR_ROUTINE (P, P, TF, LEQ_P_P),
2680 P 2800 1 OPERATOR_ROUTINE (F, F, TF, LEQ_F_F),
2681 2801 1 OPERATOR_ROUTINE (D, D, TF, LEQ_D_D));
2682 2802 1
2683 2803 1 ! Define the Operator Routine Table for COBOL NOT.
2684 2804 1
2685 P 2805 1 OPERATOR ROUTINE TABLE (COBOL_NOT_TABLE,
2686 2806 1 OPERATOR_ROUTINE (TF, TF, TF, NOT_L));
2687 2807 1
2688 2808 1 ! Define the Operator Routine Table for COBOL AND.
2689 2809 1
2690 P 2810 1 OPERATOR ROUTINE TABLE (COBOL_AND_TABLE,
2691 2811 1 OPERATOR_ROUTINE (TF, TF, TF, AND_L_L));
2692 2812 1
2693 2813 1 ! Define the Operator Routine Table for COBOL OR.
2694 2814 1
2695 P 2815 1 OPERATOR ROUTINE TABLE (COBOL_OR_TABLE,
2696 2816 1 OPERATOR_ROUTINE (TF, TF, TF, OR_L_L));
2697 2817 1
2698 2818 1
2699 2819 1 ! Define the Operator Information Table for COBOL.
2700 2820 1
2701 P 2821 1 OPERATOR INFO TABLE (COBOL_OPINFO_TABLE,
2702 P 2822 1 OPERATOR_INFO_ENTRY
2703 P 2823 1 (ADD, COBOL_ADD_TABLE, COBOL_HIER_TABLE, TABLEBASE),
2704 P 2824 1 OPERATOR_INFO_ENTRY
2705 P 2825 1 (SUBTRACT, COBOL_SUB_TABLE, COBOL_HIER_TABLE, TABLEBASE),
2706 P 2826 1 OPERATOR_INFO_ENTRY
2707 P 2827 1 (MULTIPLY, COBOL_MUL_TABLE, COBOL_HIER_TABLE, TABLEBASE),
2708 P 2828 1 OPERATOR_INFO_ENTRY
```

```
.. 2709      P 2829 1
.. 2710      P 2830 1
.. 2711      P 2831 1
.. 2712      P 2832 1
.. 2713      P 2833 1
.. 2714      P 2834 1
.. 2715      P 2835 1
.. 2716      P 2836 1
.. 2717      P 2837 1
.. 2718      P 2838 1
.. 2719      P 2839 1
.. 2720      P 2840 1
.. 2721      P 2841 1
.. 2722      P 2842 1
.. 2723      P 2843 1
.. 2724      P 2844 1
.. 2725      P 2845 1
.. 2726      P 2846 1
.. 2727      P 2847 1
.. 2728      P 2848 1
.. 2729      P 2849 1
.. 2730      P 2850 1
.. 2731      P 2851 1
.. 2732      P 2852 1
.. 2733      P 2853 1
.. 2734      P 2854 1
.. 2735      P 2855 1
.. 2736      P 2856 1
.. 2737      P 2857 1
.. 2738      P 2858 1

(DIVIDE, COBOL_DIV_TABLE, COBOL_HIER_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(UNARY_PLUS, COBOL_UNARY_PLUS_TABLE, COBOL_HIER_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(UNARY_MINUS, COBOL_UNARY_MINUS_TABLE, COBOL_HIER_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(EQUAL, COBOL_EQL_TABLE, COBOL_HIER_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(NOT_EQUAL, COBOL_NEQ_TABLE, COBOL_HIER_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(GTR_THAN, COBOL_GTR_TABLE, COBOL_HIER_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(GTR_EQUAL, COBOL_GEQ_TABLE, COBOL_HIER_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(LSS_THAN, COBOL_LSS_TABLE, COBOL_HIER_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(LSS_EQUAL, COBOL_LEQ_TABLE, COBOL_HIER_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(NOT, COBOL_NOT_TABLE, COBOL_HIER_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(AND, COBOL_AND_TABLE, COBOL_HIER_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(OR, COBOL_OR_TABLE, COBOL_HIER_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(CONVERT, TABLEBASE, COBOL_HIERD_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(DEPOSIT, TABLEBASE, COBOL_HIERD_TABLE, TABLEBASE),
OPERATOR_INFO_ENTRY
(IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE));
```

2740 2859 1  
2741 2860 1  
2742 2861 1  
2743 2862 1  
2744 2863 1  
2745 2864 1  
2746 2865 1  
2747 2866 1  
2748 2867 1  
2749 2868 1  
2750 2869 1  
2751 2870 1  
2752 2871 1  
2753 2872 1  
2754 2873 1  
2755 2874 1  
2756 2875 1  
2757 2876 1  
2758 2877 1  
2759 2878 1  
2760 2879 1  
2761 2880 1  
2762 2881 1  
2763 2882 1  
2764 2883 1  
2765 2884 1  
2766 2885 1  
2767 2886 1  
2768 2887 1  
2769 2888 1  
2770 2889 1  
2771 2890 1  
2772 2891 1  
2773 2892 1  
2774 2893 1  
2775 2894 1  
2776 2895 1  
2777 2896 1  
2778 2897 1  
2779 2898 1  
2780 2899 1  
2781 2900 1  
2782 2901 1  
2783 2902 1  
2784 2903 1  
2785 2904 1  
2786 2905 1  
2787 2906 1  
2788 2907 1  
2789 2908 1  
2790 2909 1  
2791 2910 1  
2792 2911 1  
2793 2912 1  
2794 2913 1  
2795 2914 1  
2796 2915 1

## F O R T R A N   O P E R A T O R   I N F O R M A T I O N   T A B L E S

This section contains the Operator Routine and Type tables needed to evaluate expressions in the FORTRAN language.

The following summarizes the information in the FORTRAN manual about data types, type conversions, and operators. There is further documentation within the tables below, describing exactly how we translate this into the DEBUG tables.

### Fortran Data:

Constants, Integer, Real, Character variables, Integer, Real, Character Array

### Fortran Data Types:

- . Logical
  - logical\*1 (byte), logical\*2 (word), logical\*4 (long)
- . Integer
  - integer\*2 (word), integer\*4 (long)
- . Real
  - real\*4 (f\_float), real\*8 (d\_float, g\_float), real\*16 (h\_float)
- . Complex
  - complex\*8 (f\_float), complex\*16 (d\_float, g\_float)
- . Character

### Expressions:

- . Arithmetic
  - Operand: numeric (logical, integer, real). Result: numeric values.
  - Operator: \*\* (exponentiation), \* (multiplication), / (division), + (addition, unary plus), - (subtraction, unary minus)
- . Character
  - Operand: character. Result: character values.
  - Operator: // (concatenation)
- . Relational
  - Operand: numerics or characters. Result: logical values.
  - Operator: .LT. (less than), .LE. (less than or equal to), .EQ. (equal to), .NE. (not equal to), .GT. (greater than), .GE. (greater than or equal to)
- . Logical
  - Operand: integer or logical. Result: logical values.
  - Operator: .AND., .OR., .XOR., .NEQV. (.XOR.), .EQV., .NOT.

### Type Conversion:

- . Arithmetic
  1. Operands are of the same data type, the result is also of that data type.
  2. Otherwise, Logical --> integer\*2 --> integer\*4 --> real\*4 --> real\*8 --> real\*16 --> complex\*8 --> complex\*16.

\*\*\* Note - although a chart such as the one given above  
\*\*\* does appear in the FORTRAN manual, that is not

```
2797 2916 1
2798 2917 1
2799 2918 1
2800 2919 1
2801 2920 1
2802 2921 1
2803 2922 1
2804 2923 1
2805 2924 1
2806 2925 1
2807 2926 1
2808 2927 1
2809 2928 1
2810 2929 1
2811 2930 1
2812 2931 1
2813 2932 1
2814 2933 1
2815 2934 1
2816 2935 1
2817 2936 1
2818 2937 1
2819 2938 1
2820 2939 1
2821 2940 1
2822 2941 1
2823 2942 1
2824 2943 1
2825 2944 1
2826 2945 1
2827 2946 1
2828 2947 1
2829 2948 1
2830 2949 1
2831 2950 1
2832 2951 1
2833 2952 1
2834 2953 1
2835 2954 1
2836 2955 1
2837 2956 1
2838 2957 1
2839 2958 1
2840 2959 1
2841 2960 1
2842 2961 1
2843 2962 1
2844 2963 1
2845 2964 1
2846 2965 1
2847 2966 1
2848 2967 1
2849 2968 1
2850 2969 1
2851 2970 1
2852 2971 1
2853 2972 1
```

\*\*\* exactly how things are really done. A more precise graph  
\*\*\* is:  
    B -> W -> L -> F -> D -> H  
                \      /      /  
                FC -> DC

That is, things become non-linear when complex is involved.  
For example, if a real\*16 is added to a complex\*8 then  
both are converted up to complex\*16

This second diagram is our hierarchy table.

If the user specified /G\_FLOAT then the diagram is the same  
except that G is substituted for D.

3. An operation involving a complex\*8 and a real\*16  
produces complex\*16 result.
4. Integer operations are performed on integers. (Logical is treated  
as integers)
5. Real operations are performed only on reals or combinations of  
real, integer, and logical. Integer --> real before the operation.
6. Real\*8, Real\*16 operations. Elements --> higher-precision (see 2)
7. Complex operations. (see 2, 4, 5, 6, and 3)

.Relational

1. Complex can be related only by the .EQ. and .NE.
2. Lower-ranked data type --> higher-ranked data type before the  
the comparison is made.

.Logical

1. Logical operator operates on logical, the result is logical.
2. Logical operator operates on integer, the operation is carried  
out bit-by-bit on the corresponding bits of the internal of the  
integer, the result is integer.
3. Logical operator operates on combined logical, integer, logical  
--> integer, then (see 2).

Define the Type Mapping Table for FORTRAN  
In FORTRAN, data that was declared as logical\*1, logical\*2, or logical\*4  
appears in the DST as BU, WU, and LU. However, for our purposes, we  
want to treat it as B, W, or L. We thus "map" the dtypes.

For example, if BU contains -1, and L contains 1, then we  
want EV BU+L to give 0, not 256.  
The type mapping table is thus needed to turn BU->B, WU->W, and LU->L

TYPE MAPPING TABLE (FORTRAN\_MAP\_TABLE,  
  TYPE\_GRAPH\_EDGE (BU, B),  
  TYPE\_GRAPH\_EDGE (WU, W),  
  TYPE\_GRAPH\_EDGE (LU, L),  
  0);

```

2854 2973 1
2855 2974 1
2856 2975 1 ! Define the Type Conversion Information Table for FORTRAN.
2857 2976 1 ! This points to the mapping table define above. There is no CVT_TABLE
2858 2977 1 ! specifying exceptions to the DBGSCVT_DX_DX rules.
2859 2978 1
2860 P 2979 1 CONVERSION_INFO_TABLE (FORTRAN_CVTINFO_TABLE,
2861 2980 1 CONVERSION_INFO_ENTRY (FORTRAN_MAP_TABLE, TABLEBASE));
2862 2981 1
2863 2982 1
2864 2983 1 ! Define the Type Hierarchy Table for FORTRAN.
2865 2984 1 ! This table is described above when we talk about conversion rules.
2866 2985 1 ! Note that the Incompatibility Table prevents mixing D with G types.
2867 2986 1 ! Leaving out the G edges, the graph specified by this table is:
2868 2987 1
2869 2988 1      B -> W -> L -> F -> D -> H
2870 2989 1          \      \
2871 2990 1          FC -> DC
2872 2991 1
2873 P 2992 1 TYPE_HIERARCHY_TABLE (FORTRAN_HIER1_TABLE,
2874 P 2993 1 TYPE_GRAPH_EDGE (B, W),
2875 P 2994 1 TYPE_GRAPH_EDGE (W, L),
2876 P 2995 1 TYPE_GRAPH_EDGE (L, F),
2877 P 2996 1 TYPE_GRAPH_EDGE (F, D),
2878 P 2997 1 TYPE_GRAPH_EDGE (F, G),
2879 P 2998 1 TYPE_GRAPH_EDGE (F, FC),
2880 P 2999 1 TYPE_GRAPH_EDGE (D, H),
2881 P 3000 1 TYPE_GRAPH_EDGE (D, DC),
2882 P 3001 1 TYPE_GRAPH_EDGE (G, H),
2883 P 3002 1 TYPE_GRAPH_EDGE (G, GC),
2884 P 3003 1 TYPE_GRAPH_EDGE (H, DC),
2885 P 3004 1 TYPE_GRAPH_EDGE (H, GC),
2886 P 3005 1 TYPE_GRAPH_EDGE (FC, DC),
2887 P 3006 1 TYPE_GRAPH_EDGE (FC, GC),
2888 3007 1 0);
2889 3008 1
2890 3009 1
2891 3010 1 ! Define a Type Hierarachy Table for FORTRAN
2892 3011 1 ! This is a subset of the HIER1 table, which is used for operators that
2893 3012 1 ! do not accept complex types. It would also be OK to use the HIER1 table
2894 3013 1 ! for those operators, but giving a smaller table makes the code run
2895 3014 1 ! faster.
2896 3015 1
2897 P 3016 1 TYPE_HIERARCHY_TABLE (FORTRAN_HIER2_TABLE,
2898 P 3017 1 TYPE_GRAPH_EDGE (B, W),
2899 P 3018 1 TYPE_GRAPH_EDGE (W, L),
2900 P 3019 1 TYPE_GRAPH_EDGE (L, F),
2901 P 3020 1 TYPE_GRAPH_EDGE (F, D),
2902 P 3021 1 TYPE_GRAPH_EDGE (F, G),
2903 P 3022 1 TYPE_GRAPH_EDGE (D, H),
2904 P 3023 1 TYPE_GRAPH_EDGE (G, H),
2905 3024 1 0);
2906 3025 1
2907 3026 1
2908 3027 1 ! Define a Type Hierarachy Table for FORTRAN
2909 3028 1 ! This is a subset of the HIER2 and HIER1 tables. It is used for those
2910 3029 1 ! operators that only accept integer types. It would also be OK to

```

```
2911 3030 1 ! use the HIER1 table but providing a smaller table speeds up the code.
2912 3031 1
2913 3032 1 TYPE_HIERARCHY_TABLE (FORTRAN_HIER3_TABLE,
2914 3033 1     TYPE_GRAPH_EDGE (B, W),
2915 3034 1     TYPE_GRAPH_EDGE (W, L),
2916 3035 1     0);
2917 3036 1
2918 3037 1 ! Define the Type Hierarchy Table for FORTRAN deposit.
2919 3038 1 This is a circular table which includes all types except T.
2920 3039 1 This means that any of the numeric types are convertible to any
2921 3040 1 of the other numeric types on a DEPOSIT.
2922 3041 1
2923 3042 1 TYPE_HIERARCHY_TABLE (FORTRAN_HIERD_TABLE,
2924 3043 1     TYPE_GRAPH_EDGE (B, W),
2925 3044 1     TYPE_GRAPH_EDGE (W, L),
2926 3045 1     TYPE_GRAPH_EDGE (L, F),
2927 3046 1     TYPE_GRAPH_EDGE (F, D),
2928 3047 1     TYPE_GRAPH_EDGE (D, G),
2929 3048 1     TYPE_GRAPH_EDGE (G, H),
2930 3049 1     TYPE_GRAPH_EDGE (H, FC),
2931 3050 1     TYPE_GRAPH_EDGE (FC, DC),
2932 3051 1     TYPE_GRAPH_EDGE (DC, GC),
2933 3052 1     TYPE_GRAPH_EDGE (GC, B),
2934 3053 1     0);
2935 3054 1
2936 3055 1 ! Define the Type Incompatibility Table for FORTRAN.
2937 3056 1 This prevents the user from mixing D and G types in an expression.
2938 3057 1
2939 3058 1 TYPE_INCOMP_TABLE (FORTRAN_INCOMP_TABLE,
2940 3059 1     TYPE_GRAPH_EDGE (D, G),
2941 3060 1     TYPE_GRAPH_EDGE (D, GC),
2942 3061 1     TYPE_GRAPH_EDGE (G, DC),
2943 3062 1     TYPE_GRAPH_EDGE (DC, GC),
2944 3063 1     0);
2945 3064 1
2946 3065 1 ++
2947 3066 1 Most of the arithmetic routines operate on two arguments of the same type.
2948 3067 1 That type may be B, W, L, F, D, G, H, FC, DC, GC, so we provide all
2949 3068 1 of those case indices.
2950 3069 1 --
2951 3070 1
2952 3071 1 ! Define the Operator Routine Table for FORTRAN addition.
2953 3072 1
2954 3073 1 OPERATOR_ROUTINE_TABLE (FORTRAN_ADD_TABLE,
2955 3074 1     OPERATOR_ROUTINE (B, B, B, ADD_B_B),
2956 3075 1     OPERATOR_ROUTINE (W, W, W, ADD_W_W),
2957 3076 1     OPERATOR_ROUTINE (L, L, L, ADD_L_L),
2958 3077 1     OPERATOR_ROUTINE (F, F, F, ADD_F_F),
2959 3078 1     OPERATOR_ROUTINE (D, D, D, ADD_D_D),
2960 3079 1     OPERATOR_ROUTINE (G, G, G, ADD_G_G),
2961 3080 1     OPERATOR_ROUTINE (H, H, H, ADD_H_H),
2962 3081 1     OPERATOR_ROUTINE (FC, FC, FC, ADD_FC_FC),
2963 3082 1     OPERATOR_ROUTINE (DC, DC, DC, ADD_DC_DC),
2964 3083 1     OPERATOR_ROUTINE (GC, GC, GC, ADD_GC_GC));
2965 3084 1
2966 3085 1
2967 3086 1 ! Define the Operator Routine Table for FORTRAN subtraction.
```

```
2968 3087 1 !
2969 3088 1 OPERATOR_ROUTINE TABLE (FORTRAN SUB TABLE,
2970 3089 1 OPERATOR_ROUTINE (B, B, B, SUB_B-B),
2971 3090 1 OPERATOR_ROUTINE (W, W, W, SUB_W-W),
2972 3091 1 OPERATOR_ROUTINE (L, L, L, SUB_L-L),
2973 3092 1 OPERATOR_ROUTINE (F, F, F, SUB_F-F),
2974 3093 1 OPERATOR_ROUTINE (D, D, D, SUB_D-D),
2975 3094 1 OPERATOR_ROUTINE (G, G, G, SUB_G-G),
2976 3095 1 OPERATOR_ROUTINE (H, H, H, SUB_H-H),
2977 3096 1 OPERATOR_ROUTINE (FC, FC, FC, SUB_FC_FC),
2978 3097 1 OPERATOR_ROUTINE (DC, DC, DC, SUB_DC_DC),
2979 3098 1 OPERATOR_ROUTINE (GC, GC, GC, SUB_GC_GC));
2980 3099 1
2981 100 1
2982 101 1 ! Define the Operator Routine Table for FORTRAN multiplication.
2983 102 1
2984 3103 1 OPERATOR_ROUTINE TABLE (FORTRAN MUL TABLE,
2985 3104 1 OPERATOR_ROUTINE (B, B, B, MUL_B-B),
2986 3105 1 OPERATOR_ROUTINE (W, W, W, MUL_W-W),
2987 3106 1 OPERATOR_ROUTINE (L, L, L, MUL_L-L),
2988 3107 1 OPERATOR_ROUTINE (F, F, F, MUL_F-F),
2989 3108 1 OPERATOR_ROUTINE (D, D, D, MUL_D-D),
2990 3109 1 OPERATOR_ROUTINE (G, G, G, MUL_G-G),
2991 3110 1 OPERATOR_ROUTINE (H, H, H, MUL_H-H),
2992 3111 1 OPERATOR_ROUTINE (FC, FC, FC, MUL_FC_FC),
2993 3112 1 OPERATOR_ROUTINE (DC, DC, DC, MUL_DC_DC),
2994 3113 1 OPERATOR_ROUTINE (GC, GC, GC, MUL_GC_GC));
2995 3114 1
2996 3115 1
2997 3116 1 ! Define the Operator Routine Table for FORTRAN division.
2998 3117 1
2999 3118 1 OPERATOR_ROUTINE TABLE (FORTRAN DIV TABLE,
3000 3119 1 OPERATOR_ROUTINE (B, B, B, DIV_B-B),
3001 3120 1 OPERATOR_ROUTINE (W, W, W, DIV_W-W),
3002 3121 1 OPERATOR_ROUTINE (L, L, L, DIV_L-L),
3003 3122 1 OPERATOR_ROUTINE (F, F, F, DIV_F-F),
3004 3123 1 OPERATOR_ROUTINE (D, D, D, DIV_D-D),
3005 3124 1 OPERATOR_ROUTINE (G, G, G, DIV_G-G),
3006 3125 1 OPERATOR_ROUTINE (H, H, H, DIV_H-H),
3007 3126 1 OPERATOR_ROUTINE (FC, FC, FC, DIV_FC_FC),
3008 3127 1 OPERATOR_ROUTINE (DC, DC, DC, DIV_DC_DC),
3009 3128 1 OPERATOR_ROUTINE (GC, GC, GC, DIV_GC_GC));
3010 3129 1
3011 3130 1
3012 3131 1 ! Define the Operator Routine Table for FORTRAN unary plus.
3013 3132 1
3014 3133 1 OPERATOR_ROUTINE_TABLE (FORTRAN_UNARY_PLUS_TABLE,
3015 3134 1
3016 3135 1 ! The following are not language dependent types. This is needed for DEBUG
3017 3136 1 types. For example, DEP/QUAD L= +1.
3018 3137 1
3019 3138 1 OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
3020 3139 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
3021 3140 1 OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
3022 3141 1
3023 3142 1 OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
3024 3143 1 OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
```

```
3025 P 144 1 OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
3026 P 145 1 OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
3027 P 146 1 OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
3028 P 147 1 OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
3029 P 148 1 OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
3030 P 149 1 OPERATOR_ROUTINE (FC, FC, FC, UNARY_PLUS_FC),
3031 P 150 1 OPERATOR_ROUTINE (DC, DC, DC, UNARY_PLUS_DC),
3032 P 151 1 OPERATOR_ROUTINE (GC, GC, GC, UNARY_PLUS_GC);
3033
3034 P 152 1
3035 P 153 1
3036 P 154 1 ! Define the Operator Routine Table for FORTRAN unary minus.
3037 P 155 1 OPERATOR_ROUTINE_TABLE (FORTRAN_UNARY_MINUS_TABLE,
3038 P 156 1
3039 P 157 1 ! The following are not language dependent types. This is needed for DEBUG
3040 P 158 1 types. For example, DEP/QUAD L= +1.
3041 P 159 1
3042 P 160 1 OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
3043 P 161 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
3044 P 162 1 OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
3045 P 163 1
3046 P 164 1 OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
3047 P 165 1 OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
3048 P 166 1 OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
3049 P 167 1 OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
3050 P 168 1 OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
3051 P 169 1 OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
3052 P 170 1 OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
3053 P 171 1 OPERATOR_ROUTINE (FC, FC, FC, UNARY_MINUS_FC),
3054 P 172 1 OPERATOR_ROUTINE (DC, DC, DC, UNARY_MINUS_DC),
3055 P 173 1 OPERATOR_ROUTINE (GC, GC, GC, UNARY_MINUS_GC);
3056
3057 P 174 1
3058 P 175 1
3059 P 176 1 ! Define the Operator Routine Table for FORTRAN exponentiation.
3060 P 177 1 Exponentiation has some mixed forms. For example, if you raise a
3061 P 178 1 floating number to an integer power, you do not necessarily
3062 P 179 1 want to first convert the int to float. Instead, a special
3063 P 180 1 routine indices such as POWER_F_L are provided to do the right thing here.
3064 P 181 1
3065 P 182 1 OPERATOR_ROUTINE_TABLE (FORTRAN_POWER_TABLE,
3066 P 183 1 OPERATOR_ROUTINE (W, W, W, POWER_W_W),
3067 P 184 1 OPERATOR_ROUTINE (L, L, L, POWER_L_L),
3068 P 185 1 OPERATOR_ROUTINE (F, L, F, POWER_F_L),
3069 P 186 1 OPERATOR_ROUTINE (D, L, D, POWER_D_L),
3070 P 187 1 OPERATOR_ROUTINE (G, L, G, POWER_G_L),
3071 P 188 1 OPERATOR_ROUTINE (H, L, H, POWER_H_L),
3072 P 189 1 OPERATOR_ROUTINE (FC, L, FC, POWER_FC_L),
3073 P 190 1 OPERATOR_ROUTINE (DC, L, DC, POWER_DC_L),
3074 P 191 1 OPERATOR_ROUTINE (GC, L, GC, POWER_GC_L),
3075 P 192 1 OPERATOR_ROUTINE (F, F, F, POWER_F_F),
3076 P 193 1 OPERATOR_ROUTINE (D, F, D, POWER_D_F),
3077 P 194 1 OPERATOR_ROUTINE (F, D, D, POWER_F_D),
3078 P 195 1 OPERATOR_ROUTINE (D, D, D, POWER_D_D),
3079 P 196 1 OPERATOR_ROUTINE (G, G, G, POWER_G_G),
3080 P 197 1 OPERATOR_ROUTINE (H, H, H, POWER_H_H),
3081 P 198 1 OPERATOR_ROUTINE (FC, FC, FC, POWER_FC_FC),
3082 P 199 1 OPERATOR_ROUTINE (DC, DC, DC, POWER_DC_DC),
3083 P 200 1
```

```
3082      OPERATOR_ROUTINE (GC, GC, GC, POWER_GC_GC));
3083
3084
3085      ! Define the Operator Routine Table for FORTRAN concatenate.
3086      ! Concatenate can only be done on strings.
3087
3088      P      OPERATOR_ROUTINE_TABLE (FORTRAN_CONCAT_TABLE,
3089      2207      OPERATOR_ROUTINE (T, T, T, CONCAT_T_T));
3090
3091
3092      ! Define the Operator Routine Table for FORTRAN equal.
3093      ! This operator can be done on strings as well as all the numeric
3094      ! types.
3095
3096      A      OPERATOR_ROUTINE_TABLE (FORTRAN_EQL_TABLE,
3097      2215      OPERATOR_ROUTINE (T, T, L, EQL_T_T),
3098      2216      OPERATOR_ROUTINE (B, B, L, EQL_B_B),
3099      2217      OPERATOR_ROUTINE (W, W, L, EQL_W_W),
3100      2218      OPERATOR_ROUTINE (L, L, L, EQL_L_L),
3101      2219      OPERATOR_ROUTINE (F, F, L, EQL_F_F),
3102      2220      OPERATOR_ROUTINE (D, D, L, EQL_D_D),
3103      2221      OPERATOR_ROUTINE (G, G, L, EQL_G_G),
3104      2222      OPERATOR_ROUTINE (H, H, L, EQL_H_H),
3105      2223      OPERATOR_ROUTINE (FC, FC, L, EQL_FC_FC),
3106      2224      OPERATOR_ROUTINE (DC, DC, L, EQL_DC_DC),
3107      2225      OPERATOR_ROUTINE (GC, GC, L, EQL_GC_GC));
3108
3109
3110      ! Define the Operator Routine Table for FORTRAN not equal.
3111      ! This operator can be done on strings as well as all the numeric types.
3112
3113      P      OPERATOR_ROUTINE_TABLE (FORTRAN_NEQ_TABLE,
3114      2232      OPERATOR_ROUTINE (T, T, L, NEQ_T_T),
3115      2233      OPERATOR_ROUTINE (B, B, L, NEQ_B_B),
3116      2234      OPERATOR_ROUTINE (W, W, L, NEQ_W_W),
3117      2235      OPERATOR_ROUTINE (L, L, L, NEQ_L_L),
3118      2236      OPERATOR_ROUTINE (F, F, L, NEQ_F_F),
3119      2237      OPERATOR_ROUTINE (D, D, L, NEQ_D_D),
3120      2238      OPERATOR_ROUTINE (G, G, L, NEQ_G_G),
3121      2239      OPERATOR_ROUTINE (H, H, L, NEQ_H_H),
3122      2240      OPERATOR_ROUTINE (FC, FC, L, NEQ_FC_FC),
3123      2241      OPERATOR_ROUTINE (DC, DC, L, NEQ_DC_DC),
3124      2242      OPERATOR_ROUTINE (GC, GC, L, NEQ_GC_GC));
3125
3126
3127      ! In the tables for the comparison operators,
3128      ! we allow strings to be compared, and also all the numeric types except
3129      ! for the complex floats.
3130
3131      ! Define the Operator Routine Table for FORTRAN greater than.
3132
3133      P      OPERATOR_ROUTINE_TABLE (FORTRAN_GTR_TABLE,
3134      2251      OPERATOR_ROUTINE (T, T, L, GTR_T_T),
3135      2252      OPERATOR_ROUTINE (B, B, L, GTR_B_B),
3136      2253      OPERATOR_ROUTINE (W, W, L, GTR_W_W),
3137      2254      OPERATOR_ROUTINE (L, L, L, GTR_L_L),
3138      2255      OPERATOR_ROUTINE (F, F, L, GTR_F_F),
3139      2256      OPERATOR_ROUTINE (D, D, L, GTR_D_D),
```

```
3139 P 3258 1 OPERATOR_ROUTINE (G, G, L, GTR_G_G);
3140 3259 1 OPERATOR_ROUTINE (H, H, L, GTR_H_H);
3141 3260 1
3142 3261 1
3143 3262 1 ! Define the Operator Routine Table for FORTRAN greater than or equal to.
3144 3263 1
3145 P 3264 1 OPERATOR_ROUTINE_TABLE (FORTRAN GEQ_TABLE,
3146 P 3265 1 OPERATOR_ROUTINE (T, T, L, GEQ_T_T),
3147 P 3266 1 OPERATOR_ROUTINE (B, B, L, GEQ_B_B),
3148 P 3267 1 OPERATOR_ROUTINE (W, W, L, GEQ_W_W),
3149 P 3268 1 OPERATOR_ROUTINE (L, L, L, GEQ_L_L),
3150 P 3269 1 OPERATOR_ROUTINE (F, F, L, GEQ_F_F),
3151 P 3270 1 OPERATOR_ROUTINE (D, D, L, GEQ_D_D),
3152 P 3271 1 OPERATOR_ROUTINE (G, G, L, GEQ_G_G),
3153 3272 1 OPERATOR_ROUTINE (H, H, L, GEQ_H_H);
3154 3273 1
3155 3274 1
3156 3275 1 ! Define the Operator Routine Table for FORTRAN less than.
3157 3276 1
3158 P 3277 1 OPERATOR_ROUTINE_TABLE (FORTRAN LSS_TABLE,
3159 P 3278 1 OPERATOR_ROUTINE (T, T, L, LSS_T_T),
3160 P 3279 1 OPERATOR_ROUTINE (B, B, L, LSS_B_B),
3161 P 3280 1 OPERATOR_ROUTINE (W, W, L, LSS_W_W),
3162 P 3281 1 OPERATOR_ROUTINE (L, L, L, LSS_L_L),
3163 P 3282 1 OPERATOR_ROUTINE (F, F, L, LSS_F_F),
3164 P 3283 1 OPERATOR_ROUTINE (D, D, L, LSS_D_D),
3165 P 3284 1 OPERATOR_ROUTINE (G, G, L, LSS_G_G),
3166 3285 1 OPERATOR_ROUTINE (H, H, L, LSS_H_H);
3167 3286 1
3168 3287 1
3169 3288 1 ! Define the Operator Routine Table for FORTRAN less than or equal to.
3170 3289 1
3171 P 3290 1 OPERATOR_ROUTINE_TABLE (FORTRAN LEQ_TABLE,
3172 P 3291 1 OPERATOR_ROUTINE (T, T, L, LEQ_T_T),
3173 P 3292 1 OPERATOR_ROUTINE (B, B, L, LEQ_B_B),
3174 P 3293 1 OPERATOR_ROUTINE (W, W, L, LEQ_W_W),
3175 P 3294 1 OPERATOR_ROUTINE (L, L, L, LEQ_L_L),
3176 P 3295 1 OPERATOR_ROUTINE (F, F, L, LEQ_F_F),
3177 P 3296 1 OPERATOR_ROUTINE (D, D, L, LEQ_D_D),
3178 P 3297 1 OPERATOR_ROUTINE (G, G, L, LEQ_G_G),
3179 3298 1 OPERATOR_ROUTINE (H, H, L, LEQ_H_H);
3180 3299 1
3181 3300 1
3182 3301 1 ! The logical operators .AND., .OR., .EQV., .NEQV., .NOT. can be applied
3183 3302 1 ! only to integer data types.
3184 3303 1
3185 3304 1 ! Define the Operator Routine Table for FORTRAN not.
3186 3305 1
3187 P 3306 1 OPERATOR_ROUTINE_TABLE (FORTRAN BIT_NOT_TABLE,
3188 P 3307 1 OPERATOR_ROUTINE (B, B, B, BIT_NOT_B),
3189 P 3308 1 OPERATOR_ROUTINE (W, W, W, BIT_NOT_W),
3190 3309 1 OPERATOR_ROUTINE (L, L, L, BIT_NOT_L);
3191 3310 1
3192 3311 1
3193 3312 1 ! Define the Operator Routine Table for FORTRAN and.
3194 3313 1
3195 P 3314 1 OPERATOR_ROUTINE_TABLE (FORTRAN_BIT_AND_TABLE,
```

```
3196 P 3315 1 OPERATOR_ROUTINE (B, B, B, BIT_AND_B_B),
3197 P 3316 1 OPERATOR_ROUTINE (W, W, W, BIT_AND_W_W),
3198 3317 1 OPERATOR_ROUTINE (L, L, L, BIT_AND_L_L));
3199 3318 1
3200 3319 1
3201 3320 1 ! Define the Operator Routine Table for FORTRAN or.
3202 3321 1
3203 P 3322 1 OPERATOR_ROUTINE_TABLE (FORTRAN_BIT_OR_TABLE,
3204 P 3323 1 OPERATOR_ROUTINE (B, B, B, BIT_OR_B_B),
3205 P 3324 1 OPERATOR_ROUTINE (W, W, W, BIT_OR_W_W),
3206 3325 1 OPERATOR_ROUTINE (L, L, L, BIT_OR_L_L));
3207 3326 1
3208 3327 1
3209 3328 1 ! Define the Operator Routine Table for FORTRAN xor, neqv
3210 3329 1
3211 P 3330 1 OPERATOR_ROUTINE_TABLE (FORTRAN_BIT_XOR_TABLE,
3212 P 3331 1 OPERATOR_ROUTINE (B, B, B, BIT_XOR_B_B),
3213 P 3332 1 OPERATOR_ROUTINE (W, W, W, BIT_XOR_W_W),
3214 3333 1 OPERATOR_ROUTINE (L, L, L, BIT_XOR_L_L));
3215 3334 1
3216 3335 1
3217 3336 1 ! Define the Operator Routine Table for FORTRAN eqv.
3218 3337 1
3219 P 3338 1 OPERATOR_ROUTINE_TABLE (FORTRAN_BIT_EQV_TABLE,
3220 P 3339 1 OPERATOR_ROUTINE (B, B, B, BIT_EQV_B_B),
3221 P 3340 1 OPERATOR_ROUTINE (W, W, W, BIT_EQV_W_W),
3222 3341 1 OPERATOR_ROUTINE (L, L, L, BIT_EQV_L_L));
3223 3342 1
3224 3343 1
3225 3344 1 ! Define the Operator Information Table for FORTRAN.
3226 3345 1
3227 P 3346 1 OPERATOR_INFO_TABLE (FORTRAN_OPINFO_TABLE,
3228 P 3347 1
3229 P 3348 1 ! The following are arithmetic tables that accept all numeric data types,
3230 P 3349 1 including complex. They thus go through the larger HIER1 table, and
3231 P 3350 1 need to specify an incompatibility table.
3232 P 3351 1
3233 P 3352 1 OPERATOR_INFO_ENTRY (ADD, FORTRAN_ADD_TABLE, FORTRAN_HIER1_TABLE,
3234 P 3353 1 FORTRAN_INCOMP_TABLE),
3235 P 3354 1 OPERATOR_INFO_ENTRY (SUBTRACT, FORTRAN_SUB_TABLE, FORTRAN_HIER1_TABLE,
3236 P 3355 1 FORTRAN_INCOMP_TABLE),
3237 P 3356 1 OPERATOR_INFO_ENTRY (MULTIPLY, FORTRAN_MUL_TABLE, FORTRAN_HIER1_TABLE,
3238 P 3357 1 FORTRAN_INCOMP_TABLE),
3239 P 3358 1 OPERATOR_INFO_ENTRY (DIVIDE, FORTRAN_DIV_TABLE, FORTRAN_HIER1_TABLE,
3240 P 3359 1 FORTRAN_INCOMP_TABLE),
3241 P 3360 1 OPERATOR_INFO_ENTRY (UNARY_PLUS, FORTRAN_UNARY_PLUS_TABLE,
3242 P 3361 1 FORTRAN_HIER1_TABLE, FORTRAN_INCOMP_TABLE),
3243 P 3362 1 OPERATOR_INFO_ENTRY (UNARY_MINUS, FORTRAN_UNARY_MINUS_TABLE,
3244 P 3363 1 FORTRAN_HIER1_TABLE, FORTRAN_INCOMP_TABLE),
3245 P 3364 1 OPERATOR_INFO_ENTRY (POWER_OF, FORTRAN_POWER_TABLE, FORTRAN_HIER1_TABLE,
3246 P 3365 1 FORTRAN_INCOMP_TABLE),
3247 P 3366 1
3248 P 3367 1 ! Equal/Notequal can accept all numeric types including complex and
3249 P 3368 1 thus also need the larger hierarchy table and the incompatibility
3250 P 3369 1 table.
3251 P 3370 1
3252 P 3371 1 OPERATOR_INFO_ENTRY (EQUAL, FORTRAN_EQL_TABLE, FORTRAN_HIER1_TABLE,
```

```
3253 P 3372 1
3254 P 3373 1
3255 P 3374 1
3256 P 3375 1
3257 P 3376 1
3258 P 3377 1
3259 P 3378 1
3260 P 3379 1
3261 P 3380 1
3262 P 3381 1
3263 P 3382 1
3264 P 3383 1
3265 P 3384 1
3266 P 3385 1
3267 P 3386 1
3268 P 3387 1
3269 P 3388 1
3270 P 3389 1
3271 P 3390 1
3272 P 3391 1
3273 P 3392 1
3274 P 3393 1
3275 P 3394 1
3276 P 3395 1
3277 P 3396 1
3278 P 3397 1
3279 P 3398 1
3280 P 3399 1
3281 P 3400 1
3282 P 3401 1
3283 P 3402 1
3284 P 3403 1
3285 P 3404 1
3286 P 3405 1
3287 P 3406 1
3288 P 3407 1
3289 P 3408 1
3290 P 3409 1
3291 P 3410 1
3292 P 3411 1
3293 P 3412 1
3294 P 3413 1
3295 P 3414 1
3296 P 3415 1
3297 P 3416 1
3298 P 3417 1
3299 P 3418 1
3300 P 3419 1
3301 P 3420 1
3302 P 3421 1
3303 P 3422 1
3304 P 3423 1
3305 P 3424 1
3306 P 3425 1
3307 P 3426 1
3308 P 3427 1
3309 P 3428 1

FORTRAN_INCOMP_TABLE),
OPERATOR INFO ENTRY (NOT_EQUAL, FORTRAN_NEQ_TABLE, FORTRAN_HIER1_TABLE,
FORTRAN_INCOMP_TABLE),

! The other relationals do not accept complex so they can get by with
! the smaller hierarachy table.

OPERATOR INFO ENTRY (GTR_THAN, FORTRAN_GTR_TABLE, FORTRAN_HIER2_TABLE,
FORTRAN_INCOMP_TABLE),
OPERATOR INFO ENTRY (GTR_EQUAL, FORTRAN_GEQ_TABLE, FORTRAN_HIER2_TABLE,
FORTRAN_INCOMP_TABLE),
OPERATOR INFO ENTRY (LSS_THAN, FORTRAN_LSS_TABLE, FORTRAN_HIER2_TABLE,
FORTRAN_INCOMP_TABLE),
OPERATOR INFO ENTRY (LSS_EQUAL, FORTRAN_LEQ_TABLE, FORTRAN_HIER2_TABLE,
FORTRAN_INCOMP_TABLE),

! The logical operators accept only integer quantities so they can
! use the smallest hierarchy table. They also do not need an
! incompatibility table.

OPERATOR INFO ENTRY (NOT, FORTRAN_BIT_NOT_TABLE, FORTRAN_HIER3_TABLE,
TABLEBASE),
OPERATOR INFO ENTRY (AND, FORTRAN_BIT_AND_TABLE, FORTRAN_HIER3_TABLE,
TABLEBASE),
OPERATOR INFO ENTRY (OR, FORTRAN_BIT_OR_TABLE, FORTRAN_HIER3_TABLE,
TABLEBASE),
OPERATOR INFO ENTRY (XOR, FORTRAN_BIT_XOR_TABLE, FORTRAN_HIER3_TABLE,
TABLEBASE),
OPERATOR INFO ENTRY (EQV, FORTRAN_BIT_EQV_TABLE, FORTRAN_HIER3_TABLE,
TABLEBASE),

! Concatenate accepts only string types and there are no conversions
! so we do not need a hierarchy table or an incompatibility table.

OPERATOR INFO ENTRY (CONCATENATE, FORTRAN_CONCAT_TABLE, TABLEBASE,
TABLEBASE),

! The CONVERT operator gets called to convert subscripts to integer
! type and to convert expressions in FOR, IF, WHILE, REPEAT statements
! to integer type. It can thus use the smaller HIER3 table to specify
! the rules for conversion to integer.

OPERATOR INFO ENTRY (CONVERT, TABLEBASE, FORTRAN_HIER3_TABLE,
TABLEBASE),

! The DEPOSIT operator gets called on the DEPOSIT command. It has
! its own hierarchy table which allows any numeric type to be
! converted to any other numeric type. The incompatibility table,
! however, still prevents depositing D types into G types and
! vice versa.

OPERATOR INFO ENTRY (DEPOSIT, TABLEBASE, FORTRAN_HIERD_TABLE,
FORTRAN_INCOMP_TABLE),

! The IDENTITY operator gets called at the end of an EVALUATE command
! to apply the PRIM_TO_VAL routine and then apply the appropriate
```

```

: 3310
: 3311
: 3312
: 3313
: 3314
: 3315
: 3316
: 3317
P 3429
P 3430
P 3431
P 3432
P 3433
P 3434
P 3435
P 3436

```

! type mappings. This will ensure that EV BU will print as a signed integer,  
for example.

! The identity operator does not require any tables.

OPERATOR\_INFO\_ENTRY (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE)  
);

## MACRO OPERATOR INFORMATION TABLES

This section contains the Operator Routine and Type tables needed to evaluate expressions in the MACRO language.

Although MACRO does not contain run-time expressions in the language, we provide a set of operators much the same as that provided for BLISS:

Arithmetic: + - \* / @ MOD (binary @ is the shift operator)

Relational: EQL NEQ LSS LEQ GTR GEQ LSSU LEQU GTRU GEQU

Logical: AND OR XOR EQV NOT

Bit selection: X<p,s,e>

Indirection: @

Define a Type Mapping Table for MACRO.

MACRO declares all its data to be labels, which means we get type ZI, even though we really want to treat it as integer. To simplify things, we just map ZI to integer here.

TYPE MAPPING TABLE (MACRO MAP\_TABLE,  
TYPE\_GRAPH\_EDGE (ZI, I),  
0);

Define the Type Conversion Information Table for MACRO.

We specify the Mapping Table defined above. There is no language specific table for conversions - we just use the rules in DBGSCVT\_DX\_DX

CONVERSION INFO TABLE (MACRO CVTINFO\_TABLE,  
CONVERSION\_INFO\_ENTRY (MACRO\_MAP\_TABLE, TABLEBASE));

Define the Type Hierarchy Table. All operations are done on signed longwords.

Thus we provide a path for all types to be converted to signed longwords.

Note - even though we do not get dtypes B, W, BU, WU, LU from the MACRO DST, we provide conversion paths here, in case the user is debugging a module produced by another language but with language set to MACRO.

TYPE HIERARCHY TABLE (MACRO HIER\_TABLE,  
TYPE\_GRAPH\_EDGE (BU, WU),  
TYPE\_GRAPH\_EDGE (WU, LU),  
TYPE\_GRAPH\_EDGE (LU, L),  
TYPE\_GRAPH\_EDGE (B, W),  
TYPE\_GRAPH\_EDGE (W, L),  
0);

Define the Type Hierarchy Table for DEPOSIT.

This is a circular graph which includes all types that can be obtained from calling PRIM\_TO\_VAL on a MACRO primary. (Plus some other types as well - see above note).

What this means is that

any type is convertible to any other type on a DEPOSIT.

TYPE HIERARCHY TABLE (MACRO HIERD\_TABLE,  
TYPE\_GRAPH\_EDGE (B, BU),

```
... 3376 P 3494 1 TYPE_GRAPH_EDGE (BU, W),
3377 P 3495 1 TYPE_GRAPH_EDGE (W, WU),
3378 P 3496 1 TYPE_GRAPH_EDGE (WU, L),
3379 P 3497 1 TYPE_GRAPH_EDGE (L, LU),
3380 P 3498 1 TYPE_GRAPH_EDGE (LU, VU),
3381 P 3499 1 TYPE_GRAPH_EDGE (VU, SVU),
3382 P 3500 1 TYPE_GRAPH_EDGE (SVU, SV),
3383 P 3501 1 TYPE_GRAPH_EDGE (SV, V),
3384 P 3502 1 TYPE_GRAPH_EDGE (V, ZI),
3385 P 3503 1 TYPE_GRAPH_EDGE (ZI, ZEM),
3386 P 3504 1 TYPE_GRAPH_EDGE (ZEM, B),
3387 P 3505 1 0);
3388 P 3506 1
3389 P 3507 1 ! All of the arithmetic, relational, and logical
3390 P 3508 1 ! operators just operate on longword quantities.
3391 P 3509 1
3392 P 3510 1 ! Define the Operator Routine Table for MACRO addition.
3393 P 3511 1
3394 P 3512 1 OPERATOR_ROUTINE_TABLE (MACRO_ADD_TABLE,
3395 P 3513 1 OPERATOR_ROUTINE (L, L, L, ADD_L_L));
3396 P 3514 1
3397 P 3515 1 ! Define the Operator Routine Table for MACRO subtraction.
3398 P 3516 1
3399 P 3517 1 OPERATOR_ROUTINE_TABLE (MACRO_SUB_TABLE,
3400 P 3518 1 OPERATOR_ROUTINE (L, L, L, SUB_L_L));
3401 P 3519 1
3402 P 3520 1 ! Define the Operator Routine Table for MACRO multiplication.
3403 P 3521 1
3404 P 3522 1 OPERATOR_ROUTINE_TABLE (MACRO_MUL_TABLE,
3405 P 3523 1 OPERATOR_ROUTINE (L, L, L, MUL_L_L));
3406 P 3524 1
3407 P 3525 1 ! Define the Operator Routine Table for MACRO Division.
3408 P 3526 1
3409 P 3527 1 OPERATOR_ROUTINE_TABLE (MACRO_DIV_TABLE,
3410 P 3528 1 OPERATOR_ROUTINE (L, L, L, DIV_L_L));
3411 P 3529 1
3412 P 3530 1 ! Define the Operator Routine Table for MACRO arithmetic shift.
3413 P 3531 1
3414 P 3532 1 OPERATOR_ROUTINE_TABLE (MACRO_SHIFT_TABLE,
3415 P 3533 1 OPERATOR_ROUTINE (L, L, L, SHIFT_LEFT_L_L));
3416 P 3534 1
3417 P 3535 1 ! Define the Operator Routine Table for MACRO modulus.
3418 P 3536 1
3419 P 3537 1 OPERATOR_ROUTINE_TABLE (MACRO_MOD_TABLE,
3420 P 3538 1 OPERATOR_ROUTINE (L, L, L, MOD_L_L));
3421 P 3539 1
3422 P 3540 1 ! Define the Operator Routine Table for MACRO Unary Plus.
3423 P 3541 1
3424 P 3542 1 OPERATOR_ROUTINE_TABLE (MACRO_UNARY_PLUS_TABLE,
3425 P 3543 1
3426 P 3544 1 ! The following are not language dependent types. This is needed for DEBUG
3427 P 3545 1 ! types. For example, DEP/QUAD L= +1.
3428 P 3546 1
3429 P 3547 1 OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
3430 P 3548 1 OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
3431 P 3549 1 OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
3432 P 3550 1 OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
```

```
3433 3551 1 OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
3434 3552 1 OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
3435 3553 1 OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
3436 3554 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
3437 3555 1 OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
3438 3556 1
3439 3557 1 OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L));
3440 3558 1
3441 3559 1 ! Define the Operator Routine Table for MACRO Unary Minus.
3442 3560 1
3443 3561 1 OPERATOR_ROUTINE_TABLE (MACRO_UNARY_MINUS_TABLE,
3444 3562 1
3445 3563 1 ! The following are not language dependent types. This is needed for DEBUG
3446 3564 1 ! types. For example, DEP/QUAD L= +1.
3447 3565 1
3448 3566 1 OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
3449 3567 1 OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
3450 3568 1 OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
3451 3569 1 OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
3452 3570 1 OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
3453 3571 1 OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
3454 3572 1 OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
3455 3573 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
3456 3574 1 OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
3457 3575 1
3458 3576 1 OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L));
3459 3577 1
3460 3578 1 ! Define the Operator Routine Table for MACRO Equal.
3461 3579 1
3462 3580 1 OPERATOR_ROUTINE_TABLE (MACRO_EQUAL_TABLE,
3463 3581 1 OPERATOR_ROUTINE (L, L, L, EQL_L_L));
3464 3582 1
3465 3583 1 ! Define the Operator Routine Table for MACRO Not Equal.
3466 3584 1
3467 3585 1 OPERATOR_ROUTINE_TABLE (MACRO_NOT_EQUAL_TABLE,
3468 3586 1 OPERATOR_ROUTINE (L, L, L, NEQ_L_L));
3469 3587 1
3470 3588 1 ! Define the Operator Routine Table for MACRO Less Than.
3471 3589 1
3472 3590 1 OPERATOR_ROUTINE_TABLE (MACRO_LSS_THAN_TABLE,
3473 3591 1 OPERATOR_ROUTINE (L, L, L, LSS_L_LT));
3474 3592 1
3475 3593 1 ! Define the Operator Routine Table for MACRO Greater Than.
3476 3594 1
3477 3595 1 OPERATOR_ROUTINE_TABLE (MACRO_GTR_THAN_TABLE,
3478 3596 1 OPERATOR_ROUTINE (L, L, L, GTR_L_LT));
3479 3597 1
3480 3598 1 ! Define the Operator Routine Table for MACRO Less Than or Equal.
3481 3599 1
3482 3600 1 OPERATOR_ROUTINE_TABLE (MACRO_LSS_EQUAL_TABLE,
3483 3601 1 OPERATOR_ROUTINE (L, L, L, LEQ_L_L));
3484 3602 1
3485 3603 1 ! Define the Operator Routine Table for MACRO Greater Than or Equal.
3486 3604 1
3487 3605 1 OPERATOR_ROUTINE_TABLE (MACRO_GTR_EQUAL_TABLE,
3488 3606 1 OPERATOR_ROUTINE (L, L, L, GEQ_L_L));
3489 3607 1
```

```
3490 3608 1 1 Define the Operator Routine Table for MACRO Less Than Unsigned.
3491 3609
3492 P 3610 OPERATOR_ROUTINE_TABLE (MACRO_LSSU_THAN_TABLE,
3493 3611 OPERATOR_ROUTINE (L, L, L, LSS_LU_LO));
3494 3612
3495 3613 1 1 Define the Operator Routine Table for MACRO Greater Than Unsigned.
3496 3614
3497 F 3615 OPERATOR_ROUTINE_TABLE (MACRO_GTRU_THAN_TABLE,
3498 3616 OPERATOR_ROUTINE (L, L, L, GTR_LU_LO));
3499 3617
3500 3618 1 1 Define the Operator Routine Table for MACRO Less Than or Equal Unsigned.
3501 3619
3502 P 3620 OPERATOR_ROUTINE_TABLE (MACRO_LSSU_EQUAL_TABLE,
3503 3621 OPERATOR_ROUTINE (L, L, L, LEQ_LU_LUT));
3504 3622
3505 3623 1 1 Define the Operator Routine Table for MACRO Greater Than or Equal Unsigned.
3506 3624
3507 P 3625 OPERATOR_ROUTINE_TABLE (MACRO_GTRU_EQUAL_TABLE,
3508 3626 OPERATOR_ROUTINE (L, L, L, GEQ_LU_LUT));
3509 3627
3510 3628 1 1 Define the Operator Routine Table for MACRO Bitwise And.
3511 3629
3512 P 3630 OPERATOR_ROUTINE_TABLE (MACRO_BIT_AND_TABLE,
3513 3631 OPERATOR_ROUTINE (L, L, L, BIT_AND_L_L));
3514 3632
3515 3633 1 1 Define the Operator Routine Table for MACRO Bitwise Or.
3516 3634
3517 F 3635 OPERATOR_ROUTINE_TABLE (MACRO_BIT_OR_TABLE,
3518 3636 OPERATOR_ROUTINE (L, L, L, BIT_OR_L_L));
3519 3637
3520 3638 1 1 Define the Operator Routine Table for MACRO Bitwise Xor.
3521 3639
3522 P 3640 OPERATOR_ROUTINE_TABLE (MACRO_BIT_XOR_TABLE,
3523 3641 OPERATOR_ROUTINE (L, L, L, BIT_XOR_L_L));
3524 3642
3525 3643 1 1 Define the Operator Routine Table for MACRO Bitwise Eqv.
3526 3644
3527 P 3645 OPERATOR_ROUTINE_TABLE (MACRO_BIT_EQV_TABLE,
3528 3646 OPERATOR_ROUTINE (L, L, L, BIT_EQV_L_L));
3529 3647
3530 3648 1 1 Define the Operator Routine Table for MACRO Bitwise Not.
3531 3649
3532 P 3650 OPERATOR_ROUTINE_TABLE (MACRO_BIT_NOT_TABLE,
3533 3651 OPERATOR_ROUTINE (L, L, L, BIT_NOT_L));
3534 3652
3535 3653 1 1 Define the Operator Routine Table for MACRO bit selection.
3536 3654 1 Bit selection is a little unusual. It basically just modifies
3537 3655 1 the descriptor for the address.
3538 3656
3539 3657 1 When not combined with a fetch, X<p,s,e> is equivalent to X+p/8
3540 3658 1 When combined with a fetch, @X<p,s,e> extracts s bits starting from
3541 3659 1 position p, with sign extension in e is 1.
3542 3660
3543 3661 1 See the routines DBGSBLISS_BITSELECT and DBGSBLISS_INDIRECTION,
3544 3662 1 and also the comments in the BLISS tables.
3545 3663
3546 P 3664 OPERATOR_ROUTINE_TABLE (MACRO_BITSELECT_TABLE,
```

```
3547 3665 1 OPERATOR_ROUTINE (L, L, L, BITSELECT));
3548 3666 1
3549 3667 1 ! Define the Operator Routine Table for MACRO indirection.
3550 3668 1 ! This operator is the only one that fetches its arguments, so
3551 3669 1 ! it is the only one that may see all of the possible data types.
3552 3670 1 ! The result of the fetch is always a longword.
3553 3671 1
3554 P 3672 1 OPERATOR_ROUTINE_TABLE (MACRO_INDIRECT_TABLE,
3555 P 3673 1 OPERATOR_ROUTINE (B, B, L, INDIRECT_LU),
3556 P 3674 1 OPERATOR_ROUTINE (W, W, L, INDIRECT_LU),
3557 P 3675 1 OPERATOR_ROUTINE (BU, BU, L, INDIRECT_LU),
3558 P 3676 1 OPERATOR_ROUTINE (WU, WU, L, INDIRECT_LU),
3559 P 3677 1 OPERATOR_ROUTINE (LU, LU, L, INDIRECT_LU),
3560 P 3678 1 OPERATOR_ROUTINE (V, V, L, INDIRECT_LU),
3561 P 3679 1 OPERATOR_ROUTINE (VU, VU, L, INDIRECT_LU),
3562 P 3680 1 OPERATOR_ROUTINE (SV, SV, L, INDIRECT_LU),
3563 P 3681 1 OPERATOR_ROUTINE (SVU, SVU, L, INDIRECT_LU),
3564 3682 1 OPERATOR_ROUTINE (L, L, L, INDIRECT_LO));
3565 3683 1
3566 3684 1 ! Define the Operator Information Table for MACRO.
3567 3685 1
3568 P 3686 1 OPERATOR_INFO_TABLE (MACRO_OPINFO_TABLE,
3569 P 3687 1
3570 P 3688 1 ! All of the arithmetic, logical, and relational operators use the
3571 P 3689 1 ! same hierarchy table. The FALSE fetch flag means that, like BLISS,
3572 P 3690 1 ! if the operands are not explicitly fetched, address arithmetic
3573 P 3691 1 ! is done. E.g.
3574 P 3692 1 ! EVAL A+B will add the addresses of A and B
3575 P 3693 1 ! EVAL @A+@B will add the contents of A and B
3576 P 3694 1
3577 P 3695 1 OPERATOR_INFO_ENTRY
3578 P 3696 1 (UNARY_PLUS, MACRO_UNARY_PLUS_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3579 P 3697 1 OPERATOR_INFO_ENTRY
3580 P 3698 1 (UNARY_MINUS, MACRO_UNARY_MINUS_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3581 P 3699 1 OPERATOR_INFO_ENTRY
3582 P 3700 1 (BIT_NOT, MACRO_BIT_NOT_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3583 P 3701 1 OPERATOR_INFO_ENTRY
3584 P 3702 1 (ADD, MACRO_ADD_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3585 P 3703 1 OPERATOR_INFO_ENTRY
3586 P 3704 1 (SUBTRACT, MACRO_SUB_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3587 P 3705 1 OPERATOR_INFO_ENTRY
3588 P 3706 1 (MULTIPLY, MACRO_MUL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3589 P 3707 1 OPERATOR_INFO_ENTRY
3590 P 3708 1 (DIVIDE, MACRO_DIV_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3591 P 3709 1 OPERATOR_INFO_ENTRY
3592 P 3710 1 (LEFT_SHIFT, MACRO_SHIFT_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3593 P 3711 1 OPERATOR_INFO_ENTRY
3594 P 3712 1 (EQUAL, MACRO_EQUAL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3595 P 3713 1 OPERATOR_INFO_ENTRY
3596 P 3714 1 (NOT_EQUAL, MACRO_NOT_EQUAL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3597 P 3715 1 OPERATOR_INFO_ENTRY
3598 P 3716 1 (GTR_THAN, MACRO_GTR_THAN_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3599 P 3717 1 OPERATOR_INFO_ENTRY
3600 P 3718 1 (LSS_THAN, MACRO_LSS_THAN_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3601 P 3719 1 OPERATOR_INFO_ENTRY
3602 P 3720 1 (GTR_EQUAL, MACRO_GTR_EQUAL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3603 P 3721 1 OPERATOR_INFO_ENTRY
```

```
3604 P 3722 1 (LSS_EQUAL, MACRO_LSS_EQUAL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3605 P 3723 1 OPERATOR_INFO_ENTRY
3606 P 3724 1 (REMAINDER, MACRO_MOD_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3607 P 3725 1 OPERATOR_INFO_ENTRY
3608 P 3726 1 (GTR_THAN_U, MACRO_GTRU_THAN_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3609 P 3727 1 OPERATOR_INFO_ENTRY
3610 P 3728 1 (LSS_THAN_U, MACRO_LSSU_THAN_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3611 P 3729 1 OPERATOR_INFO_ENTRY
3612 P 3730 1 (GTR_EQUAL_U, MACRO_GTRU_EQUAL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3613 P 3731 1 OPERATOR_INFO_ENTRY
3614 P 3732 1 (LSS_EQUAL_U, MACRO_LSSU_EQUAL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3615 P 3733 1 OPERATOR_INFO_ENTRY
3616 P 3734 1 (BIT_AND, MACRO_BIT_AND_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3617 P 3735 1 OPERATOR_INFO_ENTRY
3618 P 3736 1 (BIT_OR, MACRO_BIT_OR_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3619 P 3737 1 OPERATOR_INFO_ENTRY
3620 P 3738 1 (BIT_XOR, MACRO_BIT_XOR_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3621 P 3739 1 OPERATOR_INFO_ENTRY
3622 P 3740 1 (BIT_EQV, MACRO_BIT_EQV_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3623 P 3741 1 OPERATOR_INFO_ENTRY
3624 P 3742 1 (BITSELECT, MACRO_BITSELECT_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3625 P 3743 1
3626 P 3744 1 ! CONVERT gets used to convert value descriptors to integer type.
3627 P 3745 1
3628 P 3746 1 OPERATOR_INFO_ENTRY
3629 P 3747 1 (CONVERT, TABLEBASE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
3630 P 3748 1
3631 P 3749 1
3632 P 3750 1 ! The INDIRECT operator (@) is the only one that does a fetch, so the
3633 P 3751 1 ! fetch flag is set to TRUE for this operator.
3634 P 3752 1
3635 P 3753 1 OPERATOR_INFO_ENTRY
3636 P 3754 1 (INDIRECT, MACRO_INDIRECT_TABLE, MACRO_HIER_TABLE, TABLEBASE, TRUE),
3637 P 3755 1
3638 P 3756 1 ! DEPOSIT uses its own hierarchy table which allows any dtype to be
3639 P 3757 1 ! deposited into any other. The fetch flag is false, meaning no implicit
3640 P 3758 1 ! fetch is done on the right-hand-side of the deposit.
3641 P 3759 1 ! E.g. DEP A = B deposits the address of B into A;
3642 P 3760 1 ! DEP A = BB deposits the contents of B into A.
3643 P 3761 1
3644 P 3762 1 OPERATOR_INFO_ENTRY
3645 P 3763 1 (DEPOSIT, TABLEBASE, MACRO_HIERD_TABLE, TABLEBASE, FALSE),
3646 P 3764 1
3647 P 3765 1 ! The IDENTITY operator is used to turn primaries into values at
3648 P 3766 1 ! the end of an evaluate command. This ensures that EVAL A will
3649 P 3767 1 ! do exactly the same thing (go through the same code path)
3650 P 3768 1 ! as EVAL A+0 or EVAL +A
3651 P 3769 1
3652 P 3770 1 OPERATOR_INFO_ENTRY
3653 P 3771 1 (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE, FALSE)
3654 P 3772 1 );
3655 P 3773 1
```

## P A S C A L O P E R A T O R I N F O R M A T I O N T A B L E S

This section contains the Operator Routine and Type tables needed to evaluate expressions in the PASCAL language.

The following summarizes the information from the PASCAL manual.

## PASCAL Data Types:

- . Ordinal (Scalar)
  - . Integer (signed L,  $-2^{31}+1$  to  $2^{31}-1$ , unsigned L, 0 to  $2^{32}-1$ ) in binary, octal, and hexadecimal. Note: Integer value > MAXINT, that value is treated unsigned, otherwise is always treated as signed.
  - . Character (single character).
  - . Boolean (TRUE, FALSE).
  - . Enumerated (an ordered set of constant values denoted by 1 to 65535 identifiers).
  - . Subrange (a limited portion of another ordinal type for use as a distinct type).
- . Real (Scalar)
  - F\_float -0.29e-38, -1.70e38, 0.29e-38, 1.70e38
  - D\_float -0.29d-38, -1.70d38, 0.29d-38, 1.70d38
  - G\_float -0.56d-308, -0.90d308, 0.56d-308, 0.90d308
  - H\_float -0.84q-4932, -0.59q4932, 0.84q-4932, 0.59q4932
- . Structured
  - RECORD, ARRAY, VARYING OF CHAR, SET, and FILE
  - Note:
    - 1. use constructor to set constant value, except FILE
    - 2. an item of any structured type can be packed, except VARYING
  - RECORD, RECORD with Variants, ARRAY, VARYING OF CHAR, SET, FILE (of ordinal, real, structured and pointer).
- . Pointer
  - dynamic variable, NIL.

## Compatibility:

- . Structural
- . Assignment

## Type Conversions:

- . Arithmetic Types
  - INTEGER (lowest)
  - UNSIGNED
  - REAL
  - DOUBLE
  - QUAD (highest)
- . Character Types
  - CHAR (lowest)
  - PACKED ARRAY OF CHAR
  - VARYING OF CHAR (highest)

```
3714 3831 1 Operators
3715 3832 1   .Arithmetic (+ - * ** / DIV REM MOD)
3716 3833 1   + - * ** operate on Arithmetic Types and produces a result of the
3717 3834 1   same type as the values.
3718 3835 1   / operates on Arithmetic Types and produces a Real result.
3719 3836 1   DIV REM MOD operate on integer and unsigned and produces an integer or
3720 3837 1   unsigned result.
3721 3838 1
3722 3839 1   .Relational (= <> < <= > >=)
3723 3840 1   Test Ordinal, Real, String or Set expressions and return a Boolean
3724 3841 1   result.
3725 3842 1
3726 3843 1   .Logical (AND OR NOT)
3727 3844 1   Operate on Boolean expressions and produce a Boolean Value.
3728 3845 1
3729 3846 1   .String (+ = <> < <= > >=)
3730 3847 1   Operate on String expressions.
3731 3848 1
3732 3849 1   .Set (+ * - = <> <= >= IN)
3733 3850 1   Form the union, intersection, or difference of two sets, compares two
3734 3851 1   sets, or tests an ordinal value for inclusion in a set.
3735 3852 1
3736 3853 1
3737 3854 1
3738 3855 1   Define a Mapping Table for PASCAL.
3739 3856 1   The compiler gives us a dtype of DST$K_BOOL.
3740 3857 1   We've defined a DSC$K_DTYPE_BOOL = DST$K_BOOL. But we then
3741 3858 1   map this dtype into our new DSC$K_DTYPE_TF. This keeps the dtype codes
3742 3859 1   dense and also compatible across languages.
3743 3860 1
3744 3861 1 TYPE MAPPING TABLE (PASCAL_MAP_TABLE,
3745 3862 1   TYPE_GRAPH_EDGE (TPTR, LU),
3746 3863 1   TYPE_GRAPH_EDGE (BOOL, TF));
3747 3864 1
3748 3865 1   Define the Type Conversion Information Table for PASCAL.
3749 3866 1   PASCAL has a mapping table but no language-specific table of
3750 3867 1   exceptions. It thus uses the DEBUG conversions given in DBG$CVT_DX_DX.
3751 3868 1
3752 3869 1 P CONVERSION INFO TABLE (PASCAL_CVTINFO_TABLE,
3753 3870 1   CONVERSION_INFO_ENTRY (PASCAL_MAP_TABLE, TABLEBASE, TRUE));
3754 3871 1
3755 3872 1   Define the Type Hierarchy Table for PASCAL.
3756 3873 1   This specifies that, for numeric types,
3757 3874 1   implicit conversions always go up along
3758 3875 1   the hierarchy
3759 3876 1
3760 3877 1   B -> M -> L -> LU -> F -> D -> H
3761 3878 1
3762 3879 1   If /G_FLOW was specified then substitute G for D in the above.
3763 3880 1   Note - the PASCAL compiler never generates B or M, but we allow
3764 3881 1   a conversion path so that such data can be accessed if your language
3765 3882 1   is set to PASCAL in a mixed-language environment.
3766 3883 1
3767 3884 1   Text can be converted to varying text for string operations.
3768 3885 1
3769 3886 1 P TYPE HIERARCHY TABLE (PASCAL_HIER_TABLE,
3770 3887 1   TYPE_GRAPH_EDGE (T, VT),
```

```
3771 P 3888 1 TYPE_GRAPH_EDGE (B, W),
3772 P 3889 1 TYPE_GRAPH_EDGE (W, L),
3773 P 3890 1 TYPE_GRAPH_EDGE (L, LU),
3774 P 3891 1 TYPE_GRAPH_EDGE (LU, F),
3775 P 3892 1 TYPE_GRAPH_EDGE (F, D),
3776 P 3893 1 TYPE_GRAPH_EDGE (F, G),
3777 P 3894 1 TYPE_GRAPH_EDGE (D, H),
3778 P 3895 1 TYPE_GRAPH_EDGE (G, H),
3779 P 3896 1 O);
3780 P 3897 1
3781 P 3898 1 ! Define a smaller Type Hierarchy Table for PASCAL.
3782 P 3899 1 ! This is a subset of the above table, and can be used for operators
3783 P 3900 1 ! that only allow integer types.
3784 P 3901 1
3785 P 3902 1 TYPE_HIERARCHY_TABLE (PASCAL_HIER1_TABLE,
3786 P 3903 1 TYPE_GRAPH_EDGE (B, W),
3787 P 3904 1 TYPE_GRAPH_EDGE (W, L),
3788 P 3905 1 TYPE_GRAPH_EDGE (L, LU),
3789 P 3906 1 O);
3790 P 3907 1
3791 P 3908 1 ! Define the Type Hierarchy Table for PASCAL deposit.
3792 P 3909 1 ! This specifies what we will allow on a DEPOSIT command. That is,
3793 P 3910 1 ! if there is a path from one dtype to another in the table below
3794 P 3911 1 ! then we will allow the first to be deposited into the second.
3795 P 3912 1
3796 P 3913 1 ! We have adopted similar rules as the PASCAL compiler; that is, in
3797 P 3914 1 ! general, if the compiler allows the assignment we will allow the deposit.
3798 P 3915 1 ! There may be a few cases where we are more permissive than the compiler.
3799 P 3916 1
3800 P 3917 1 TYPE_HIERARCHY_TABLE (PASCAL_HIERD_TABLE,
3801 P 3918 1 TYPE_GRAPH_EDGE (L, LU),
3802 P 3919 1
3803 P 3920 1 ! Allow integers to be deposited into any of SUBRNG, or ENUM.
3804 P 3921 1 ! The type converter will call a TYPEID_CHECK routine which will
3805 P 3922 1 ! give an informational if the language does not allow the
3806 P 3923 1 ! assignment. We do, however, do the operation anyway.
3807 P 3924 1
3808 P 3925 1 TYPE_GRAPH_EDGE (LU, SUBRNG),
3809 P 3926 1 TYPE_GRAPH_EDGE (LU, ENUM),
3810 P 3927 1 TYPE_GRAPH_EDGE (LU, SET),
3811 P 3928 1
3812 P 3929 1 ! Allow assignments in an upward direction.
3813 P 3930 1
3814 P 3931 1 TYPE_GRAPH_EDGE (LU, F),
3815 P 3932 1 TYPE_GRAPH_EDGE (F, D),
3816 P 3933 1 TYPE_GRAPH_EDGE (F, G),
3817 P 3934 1 TYPE_GRAPH_EDGE (D, H),
3818 P 3935 1 TYPE_GRAPH_EDGE (G, H),
3819 P 3936 1
3820 P 3937 1 ! Text can be assigned into a VT variable.
3821 P 3938 1
3822 P 3939 1 TYPE_GRAPH_EDGE (T, VT),
3823 P 3940 1
3824 P 3941 1 ! Allow assignments of T, ENUM, and TF to subrange. Subrange
3825 P 3942 1 ! can be such things as 'a'..'z', red..green, or false..true
3826 P 3943 1
3827 P 3944 1 TYPE_GRAPH_EDGE (T, SET),
```

```
3828 P 3945 1 TYPE_GRAPH_EDGE (T, SUBRNG),
3829 P 3946 1 TYPE_GRAPH_EDGE (ENUM, SET),
3830 P 3947 1 TYPE_GRAPH_EDGE (ENUM, SUBRNG),
3831 P 3948 1 TYPE_GRAPH_EDGE (TF, SET),
3832 3949 1 TYPE_GRAPH_EDGE (TF, SUBRNG));
3833 3950 1
3834 3951 1
3835 3952 1 ! Define the Type Incompatibility Table for PASCAL
3836 3953 1 ! We do not allow types D and G to be mixed in arithmetic statements.
3837 3954 1
3838 P 3955 1 TYPE_INCOMP_TABLE (PASCAL_INCOMP_TABLE,
3839 P 3956 1 TYPE_GRAPH_EDGE (D, G),
3840 3957 1 0);
3841 3958 1
3842 3959 1
3843 3960 1 ! Define the Operator Routine Table for PASCAL Addition.
3844 3961 1 ! We can add any of the numeric types. "+" can also be applied to sets,
3845 3962 1 ! in which case it means set union. It can be applied to varying strings
3846 3963 1 ! and in this case it means string concatenation.
3847 3964 1 ! The additional specifier for strings and sets refers to a typeid check
3848 3965 1 ! routine that checks that the arguments are really of compatible type.
3849 3966 1 ! See the routine DBG$PERFORM_TYPEID_CHECK in DBGEVALOP.
3850 3967 1
3851 P 3968 1 OPERATOR_ROUTINE_TABLE (PASCAL_ADD_TABLE,
3852 P 3969 1 OPERATOR_ROUTINE (VT, VT, VT, CONCAT_T_T),
3853 P 3970 1 OPERATOR_ROUTINE (SET, SET, SET, UNION_SET_SET, SET_SET),
3854 P 3971 1 OPERATOR_ROUTINE (L, L, L, ADD_L_L),
3855 P 3972 1 OPERATOR_ROUTINE (LU, LU, LU, ADD_LU_LU),
3856 P 3973 1 OPERATOR_ROUTINE (F, F, F, ADD_F_F),
3857 P 3974 1 OPERATOR_ROUTINE (D, D, D, ADD_D_D),
3858 P 3975 1 OPERATOR_ROUTINE (G, G, G, ADD_G_G),
3859 3976 1 OPERATOR_ROUTINE (H, H, H, ADD_H_H));
3860 3977 1
3861 3978 1
3862 3979 1 ! Define the Operator Routine Table for PASCAL Subtraction.
3863 3980 1 ! Subtraction can be done to any of the numeric type.
3864 3981 1 ! "-" can also be applied to sets, in which case it means set difference.
3865 3982 1 ! The additional specifier for sets refers to a typeid check
3866 3983 1 ! routine that checks that the arguments are really of compatible type.
3867 3984 1 ! See the routine DBG$PERFORM_TYPEID_CHECK in DBGEVALOP.
3868 3985 1
3869 P 3986 1 OPERATOR_ROUTINE_TABLE (PASCAL_SUB_TABLE,
3870 P 3987 1 OPERATOR_ROUTINE (SET, SET, SET, DIFFERENCE_SET_SET, SET_SET),
3871 P 3988 1 OPERATOR_ROUTINE (L, L, L, SUB_L_L),
3872 P 3989 1 OPERATOR_ROUTINE (LU, LU, LU, SUB_LU_LU),
3873 P 3990 1 OPERATOR_ROUTINE (F, F, F, SUB_F_F),
3874 P 3991 1 OPERATOR_ROUTINE (D, D, D, SUB_D_D),
3875 P 3992 1 OPERATOR_ROUTINE (G, G, G, SUB_G_G),
3876 3993 1 OPERATOR_ROUTINE (H, H, H, SUB_H_H));
3877 3994 1
3878 3995 1
3879 3996 1 ! Define the Operator Routine Table for PASCAL Multiplication.
3880 3997 1 ! Multiplication can be applied to any of the numeric types.
3881 3998 1 ! "*" can also be applied to sets, in which case it means set difference.
3882 3999 1 ! The additional specifier for sets refers to a typeid check
3883 4000 1 ! routine that checks that the arguments are really of compatible type.
3884 4001 1 ! See the routine DBG$PERFORM_TYPEID_CHECK in DBGEVALOP.
```

```
3885 4002 1 !
3886 P 4003 1 OPERATOR_ROUTINE_TABLE (PASCAL_MUL_TABLE,
3887 P 4004 1 OPERATOR_ROUTINE (SET, SET, SET, INTERSECT_SET_SET, SET_SET),
3888 P 4005 1 OPERATOR_ROUTINE (L, L, L, MUL_L_L),
3889 P 4006 1 OPERATOR_ROUTINE (LU, LU, LU, MUL_LU_LU),
3890 P 4007 1 OPERATOR_ROUTINE (F, F, F, MUL_F_F),
3891 P 4008 1 OPERATOR_ROUTINE (D, D, D, MUL_D_D),
3892 P 4009 1 OPERATOR_ROUTINE (G, G, G, MUL_G_G),
3893 4010 1 OPERATOR_ROUTINE (H, H, H, MUL_H_H));
3894 4011 1
3895 4012 1
3896 4013 1 ! Define the Operator Routine Table for PASCAL Exponentiation.
3897 4014 1 ! This can be applied to any of the numeric types.
3898 4015 1
3899 P 4016 1 OPERATOR_ROUTINE_TABLE (PASCAL_POWER_TABLE,
3900 P 4017 1 OPERATOR_ROUTINE (L, L, L, POWER_L_L),
3901 P 4018 1 OPERATOR_ROUTINE (LU, LU, LU, POWER_LU_L),
3902 P 4019 1 OPERATOR_ROUTINE (F, F, F, POWER_F_F),
3903 P 4020 1 OPERATOR_ROUTINE (D, D, D, POWER_D_D),
3904 P 4021 1 OPERATOR_ROUTINE (G, G, G, POWER_G_G),
3905 4022 1 OPERATOR_ROUTINE (H, H, H, POWER_H_H));
3906 4023 1
3907 4024 1
3908 4025 1 ! Define the Operator Routine Table for PASCAL Division.
3909 4026 1 ! PASCAL "/" is only for floating division, "DIV" is used for integer
3910 4027 1 ! division.
3911 4028 1
3912 P 4029 1 OPERATOR_ROUTINE_TABLE (PASCAL_DIV_TABLE,
3913 P 4030 1 OPERATOR_ROUTINE (F, F, F, DIV_F_F),
3914 P 4031 1 OPERATOR_ROUTINE (D, D, D, DIV_D_D),
3915 P 4032 1 OPERATOR_ROUTINE (G, G, G, DIV_G_G),
3916 4033 1 OPERATOR_ROUTINE (H, H, H, DIV_H_H));
3917 4034 1
3918 4035 1 ! Unary Plus and Minus can be applied to any of the numeric types.
3919 4036 1
3920 4037 1 ! Define the Operator Routine Table for PASCAL Unary plus.
3921 4038 1
3922 P 4039 1 OPERATOR_ROUTINE_TABLE (PASCAL_UNARY_PLUS_TABLE,
3923 P 4040 1
3924 P 4041 1 ! The following are not language dependent types. This is needed for DEBUG
3925 P 4042 1 ! types. For example, DEP/QUAD L= +1.
3926 P 4043 1 !
3927 P 4044 1 OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
3928 P 4045 1 OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
3929 P 4046 1 OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
3930 P 4047 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
3931 P 4048 1 OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
3932 P 4049 1
3933 P 4050 1 OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
3934 P 4051 1 OPERATOR_ROUTINE (LU, LU, LU, UNARY_PLUS_LU),
3935 P 4052 1 OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
3936 P 4053 1 OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
3937 P 4054 1 OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
3938 4055 1 OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H);
3939 4056 1
3940 4057 1
3941 4058 1 ! Define the Operator Routine Table for PASCAL unary minus.
```

```

3942 4059 1 !
3943 P 4060 1 OPERATOR_ROUTINE_TABLE (PASCAL_UNARY_MINUS_TABLE,
3944 P 4061 1
3945 P 4062 1 ! The following are not language dependent types. This is needed for DEBUG
3946 P 4063 1 ! types. For example, DEP/QUAD L= +1.
3947 P 4064 1 !
3948 P 4065 1 OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
3949 P 4066 1 OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
3950 P 4067 1 OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
3951 P 4068 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
3952 P 4069 1 OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
3953 P 4070 1
3954 P 4071 1 OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
3955 P 4072 1 OPERATOR_ROUTINE (LU, LU, LU, UNARY_MINUS_L),
3956 P 4073 1 OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
3957 P 4074 1 OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
3958 P 4075 1 OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
3959 4076 1 OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H));
3960 4077 1
3961 4078 1
3962 4079 1 ! Define the Operator Routine Table for PASCAL Integer Division.
3963 4080 1 ! PASCAL has a DIV operator for integer divide.
3964 4081 1 !
3965 P 4082 1 OPERATOR_ROUTINE_TABLE (PASCAL_INTDIV_TABLE,
3966 P 4083 1 OPERATOR_ROUTINE (L, L, L, DIV_L_L),
3967 4084 1 OPERATOR_ROUTINE (LU, LU, LU, DIV_LU_LU));
3968 4085 1
3969 4086 1 ! MOD and REM can only be applied to integer arguments.
3970 4087 1
3971 4088 1 ! Define the Operator Routine Table for PASCAL Modulus.
3972 4089 1 !
3973 P 4090 1 OPERATOR_ROUTINE_TABLE (PASCAL_MOD_TABLE,
3974 P 4091 1 OPERATOR_ROUTINE (L, L, L, MOD_L_L),
3975 4092 1 OPERATOR_ROUTINE (LU, LU, LU, MOD_LU_LU));
3976 4093 1
3977 4094 1
3978 4095 1 ! Define the Operator Routine Table for PASCAL Remainder.
3979 4096 1 !
3980 P 4097 1 OPERATOR_ROUTINE_TABLE (PASCAL_REM_TABLE,
3981 P 4098 1 OPERATOR_ROUTINE (L, L, L, REM_L_L),
3982 4099 1 OPERATOR_ROUTINE (LU, LU, LU, REM_LU_LU));
3983 4100 1
3984 4101 1
3985 4102 1 ! Equal/Not equal can be applied to numeric types, sets, enumeration types,
3986 4103 1 ! strings, pointers, and Booleans.
3987 4104 1 ! The SET_SET, and ENUM_ENUM parameters below
3988 4105 1 ! specify a typeid check to be done to ensure that operands being compared
3989 4106 1 ! are really of compatible types. See the DBG$PERFORM_TYPEID_CHECK routine
3990 4107 1 ! in DBGVALOP.
3991 4108 1
3992 4109 1 ! Define the Operator Routine Table for PASCAL equal.
3993 4110 1 !
3994 P 4111 1 OPERATOR_ROUTINE_TABLE (PASCAL_EQL_TABLE,
3995 P 4112 1 OPERATOR_ROUTINE (SET, SET, TF, EQL_SET_SET, SET_SET),
3996 P 4113 1 OPERATOR_ROUTINE (TF, TF, TF, EQL_L_L),
3997 P 4114 1 OPERATOR_ROUTINE (ENUM, ENUM, TF, EQL_L_L, ENUM_ENUM),
3998 P 4115 1 OPERATOR_ROUTINE (VT, VT, TF, EQL_VT_VT),

```

```
3999 P 4116 1 OPERATOR_ROUTINE (L, L, TF, EQL_L_L),
4000 P 4117 1 OPERATOR_ROUTINE (LU, LU, TF, EQL_L_L),
4001 P 4118 1 OPERATOR_ROUTINE (F, F, TF, EQL_F_F),
4002 P 4119 1 OPERATOR_ROUTINE (D, D, TF, EQL_D_D),
4003 P 4120 1 OPERATOR_ROUTINE (G, G, TF, EQL_G_G),
4004 4121 1 OPERATOR_ROUTINE (H, H, TF, EQL_H_H);
4005 4122 1
4006 4123 1
4007 4124 1 ! Define the Operator Routine Table for PASCAL not equal.
4008 4125 1
4009 P 4126 1 OPERATOR_ROUTINE_TABLE (PASCAL_NEQ_TABLE,
4010 P 4127 1 OPERATOR_ROUTINE (SET, SET, TF, NEQ_SET_SET, SET_SET),
4011 P 4128 1 OPERATOR_ROUTINE (TF, TF, TF, NEQ_L_L),
4012 P 4129 1 OPERATOR_ROUTINE (ENUM, ENUM, TF, NEQ_L_L, ENUM_ENUM),
4013 P 4130 1 OPERATOR_ROUTINE (VT, VT, TF, NEQ_VT_VT),
4014 P 4131 1 OPERATOR_ROUTINE (L, L, TF, NEQ_L_L),
4015 P 4132 1 OPERATOR_ROUTINE (LU, LU, TF, NEQ_L_L),
4016 P 4133 1 OPERATOR_ROUTINE (F, F, TF, NEQ_F_F),
4017 P 4134 1 OPERATOR_ROUTINE (D, D, TF, NEQ_D_D),
4018 P 4135 1 OPERATOR_ROUTINE (G, G, TF, NEQ_G_G),
4019 4136 1 OPERATOR_ROUTINE (H, H, TF, NEQ_H_H);
4020 4137 1
4021 4138 1
4022 4139 1 ! Define the Operator Routine Table for PASCAL greater than or equal to.
4023 4140 1 ! Greater/equal can be applied to numeric types, sets, enumeration types,
4024 4141 1 ! strings, and Booleans.
4025 4142 1 ! On sets, it means "contains".
4026 4143 1
4027 P 4144 1 OPERATOR_ROUTINE_TABLE (PASCAL_GEQ_TABLE,
4028 P 4145 1 OPERATOR_ROUTINE (SET, SET, TF, GEQ_SET_SET, SET_SET),
4029 P 4146 1 OPERATOR_ROUTINE (TF, TF, TF, GEQ_L_L),
4030 P 4147 1 OPERATOR_ROUTINE (ENUM, ENUM, TF, GEQ_L_L, ENUM_ENUM),
4031 P 4148 1 OPERATOR_ROUTINE (VT, VT, TF, GEQ_VT_VT),
4032 P 4149 1 OPERATOR_ROUTINE (L, L, TF, GEQ_L_L),
4033 P 4150 1 OPERATOR_ROUTINE (LU, LU, TF, GEQ_LU_LU),
4034 P 4151 1 OPERATOR_ROUTINE (F, F, TF, GEQ_F_F),
4035 P 4152 1 OPERATOR_ROUTINE (D, D, TF, GEQ_D_D),
4036 P 4153 1 OPERATOR_ROUTINE (G, G, TF, GEQ_G_G),
4037 4154 1 OPERATOR_ROUTINE (H, H, TF, GEQ_H_H);
4038 4155 1
4039 4156 1
4040 4157 1 ! Define the Operator Routine Table for PASCAL great than.
4041 4158 1 ! Greater than can be applied to numeric types, enumeration types,
4042 4159 1 ! strings, and Booleans.
4043 4160 1
4044 P 4161 1 OPERATOR_ROUTINE_TABLE (PASCAL_GTR_TABLE,
4045 P 4162 1 OPERATOR_ROUTINE (TF, TF, TF, GTR_L_L),
4046 P 4163 1 OPERATOR_ROUTINE (ENUM, ENUM, TF, GTR_L_L, ENUM_ENUM),
4047 P 4164 1 OPERATOR_ROUTINE (VT, VT, TF, GTR_VT_VT),
4048 P 4165 1 OPERATOR_ROUTINE (L, L, TF, GTR_L_L),
4049 P 4166 1 OPERATOR_ROUTINE (LU, LU, TF, GTR_LU_LU),
4050 P 4167 1 OPERATOR_ROUTINE (F, F, TF, GTR_F_F),
4051 P 4168 1 OPERATOR_ROUTINE (D, D, TF, GTR_D_D),
4052 P 4169 1 OPERATOR_ROUTINE (G, G, TF, GTR_G_G),
4053 4170 1 OPERATOR_ROUTINE (H, H, TF, GTR_H_H);
4054 4171 1
4055 4172 1
```

```
4056      4173 1
4057      4174 1
4058      4175 1 ! Define the Operator Routine Table for PASCAL less than.
4059      4176 1 ! Less than can be applied to numeric types, enumeration types,
4060      4177 1 ! strings, and Booleans.
4061      4178 1
4062      P 4179 1 OPERATOR_ROUTINE TABLE (PASCAL_LSS_TABLE,
4063      P 4180 1   OPERATOR_ROUTINE (TF, TF, TF, LSS_L_L),
4064      P 4181 1   OPERATOR_ROUTINE (ENUM, ENUM, TF, LSS_L_L, ENUM_ENUM),
4065      P 4182 1   OPERATOR_ROUTINE (VT, VT, TF, LSS_VT_VT),
4066      P 4183 1   OPERATOR_ROUTINE (L, L, TF, LSS_L_L),
4067      P 4184 1   OPERATOR_ROUTINE (LU, LU, TF, LSS_LU_LU),
4068      P 4185 1   OPERATOR_ROUTINE (F, F, TF, LSS_F_F),
4069      P 4186 1   OPERATOR_ROUTINE (D, D, TF, LSS_D_D),
4070      P 4187 1   OPERATOR_ROUTINE (G, G, TF, LSS_G_G),
4071      4188 1   OPERATOR_ROUTINE (H, H, TF, LSS_H_H);
4072      4189 1
4073      4190 1
4074      4191 1 ! Define the Operator Routine Table for PASCAL less than or equal to.
4075      4192 1 ! Less/equal can be applied to numeric types, sets, enumeration types,
4076      4193 1 ! strings, and Booleans.
4077      4194 1 ! On sets it means "is contained in".
4078      4195 1
4079      P 4196 1 OPERATOR_ROUTINE TABLE (PASCAL_LEQ_TABLE,
4080      P 4197 1   OPERATOR_ROUTINE (SET, SET, TF, LEQ_SET_SET, SET_SET),
4081      P 4198 1   OPERATOR_ROUTINE (TF, TF, TF, LEQ_L_L),
4082      P 4199 1   OPERATOR_ROUTINE (ENUM, ENUM, TF, LEQ_L_L, ENUM_ENUM),
4083      P 4200 1   OPERATOR_ROUTINE (VT, VT, TF, LEQ_VT_VT),
4084      P 4201 1   OPERATOR_ROUTINE (L, L, TF, LEQ_L_L),
4085      P 4202 1   OPERATOR_ROUTINE (LU, LU, TF, LEQ_LU_LU),
4086      P 4203 1   OPERATOR_ROUTINE (F, F, TF, LEQ_F_F),
4087      P 4204 1   OPERATOR_ROUTINE (D, D, TF, LEQ_D_D),
4088      P 4205 1   OPERATOR_ROUTINE (G, G, TF, LEQ_G_G),
4089      4206 1   OPERATOR_ROUTINE (H, H, TF, LEQ_H_H);
4090      4207 1
4091      4208 1
4092      4209 1 ! The logical operators can only be applied to Booleans.
4093      4210 1
4094      4211 1 ! Define the Operator Routine Table for PASCAL not.
4095      4212 1
4096      P 4213 1 OPERATOR_ROUTINE TABLE (PASCAL_NOT_TABLE,
4097      4214 1   OPERATOR_ROUTINE (TF, TF, TF, NOT_L));
4098      4215 1
4099      4216 1
4100      4217 1 ! Define the Operator Routine Table for PASCAL and.
4101      4218 1
4102      P 4219 1 OPERATOR_ROUTINE TABLE (PASCAL_AND_TABLE,
4103      4220 1   OPERATOR_ROUTINE (TF, TF, TF, AND_L_L));
4104      4221 1
4105      4222 1
4106      4223 1 ! Define the Operator Routine Table for PASCAL or.
4107      4224 1
4108      P 4225 1 OPERATOR_ROUTINE TABLE (PASCAL_OR_TABLE,
4109      4226 1   OPERATOR_ROUTINE (TF, TF, TF, OR_L_L));
4110      4227 1
4111      4228 1
4112      4229 1 ! Define the Operator Routine Table for PASCAL Set IN.
```

```
4113 4230 1 ! Sets can be composed of integers, characters, Booleans, or
4114 4231 1 ! enumeration types, so we allow the inquiry whether an element
4115 4232 1 ! of one of those types is in the set. The typeid check routine
4116 4233 1 ! is present to check that the type of the left arg matches the
4117 4234 1 ! element type of the set.
4118 4235 1
4119 P 4236 1 OPERATOR_ROUTINE_TABLE (PASCAL_IN_TABLE,
4120 P 4237 1 OPERATOR_ROUTINE (L, SET, TF, IN SET SET, SET SET),
4121 P 4238 1 OPERATOR_ROUTINE (LU, SET, TF, IN SET SET, SET SET),
4122 P 4239 1 OPERATOR_ROUTINE (T, SET, TF, IN SET SET, SET SET),
4123 P 4240 1 OPERATOR_ROUTINE (TF, SET, TF, IN SET SET, SET SET),
4124 4241 1 OPERATOR_ROUTINE (ENUM, SET, TF, IN SET SET, SET SET));
4125 4242 1
4126 4243 1
4127 4244 1 ! Define the Operator Routine Table for the PASCAL built-in function SUCC.
4128 4245 1
4129 P 4246 1 OPERATOR_ROUTINE_TABLE (PASCAL_SUCCESOR_TABLE,
4130 P 4247 1 OPERATOR_ROUTINE (ENUM, ENUM, ENUM, SUCC_ENUM, ENUM_ENUM));
4131 4248 1
4132 4249 1
4133 4250 1 ! Define the Operator Routine Table for the PASCAL built-in function PRED.
4134 4251 1
4135 P 4252 1 OPERATOR_ROUTINE_TABLE (PASCAL_PREDECESSOR_TABLE,
4136 4253 1 OPERATOR_ROUTINE (ENUM, ENUM, ENUM, PRED_ENUM, ENUM_ENUM));
4137 4254 1
4138 4255 1
4139 4256 1 ! Define the Operator Information Table for PASCAL.
4140 4257 1
4141 P 4258 1 OPERATOR_INFO_TABLE (PASCAL_OPINFO_TABLE,
4142 P 4259 1
4143 P 4260 1 ! The following arithmetic operators use the normal hierarchy table and
4144 P 4261 1 ! the incompatibility table.
4145 P 4262 1
4146 P 4263 1 OPERATOR_INFO_ENTRY
4147 P 4264 1 (ADD, PASCAL_ADD_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4148 P 4265 1 OPERATOR_INFO_ENTRY
4149 P 4266 1 (SUBTRACT, PASCAL_SUB_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4150 P 4267 1 OPERATOR_INFO_ENTRY
4151 P 4268 1 (MULTIPLY, PASCAL_MUL_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4152 P 4269 1 OPERATOR_INFO_ENTRY
4153 P 4270 1 (POWER_OF, PASCAL_POWER_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4154 P 4271 1 OPERATOR_INFO_ENTRY
4155 P 4272 1 (DIVIDE, PASCAL_DIV_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4156 P 4273 1 OPERATOR_INFO_ENTRY
4157 P 4274 1 (UNARY_PLUS, PASCAL_UNARY_PLUS_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4158 P 4275 1 OPERATOR_INFO_ENTRY
4159 P 4276 1 (UNARY_MINUS, PASCAL_UNARY_MINUS_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4160 P 4277 1
4161 P 4278 1 ! DIV, MOD, and REM operate only on integers and thus use the smaller
4162 P 4279 1 ! hierarchy table HIER1. It would also be OK to use the HIER table
4163 P 4280 1 ! but a smaller table makes the code run faster.
4164 P 4281 1
4165 P 4282 1 OPERATOR_INFO_ENTRY
4166 P 4283 1 (INT_DIVIDE, PASCAL_INTDIV_TABLE, PASCAL_HIER1_TABLE, TABLEBASE),
4167 P 4284 1 OPERATOR_INFO_ENTRY
4168 P 4285 1 (MODULUS, PASCAL_MCO_TABLE, PASCAL_HIER1_TABLE, TABLEBASE),
4169 P 4286 1 OPERATOR_INFO_ENTRY
```

```
4170 P 4287 1 (REMAINDER, PASCAL_REM_TABLE, PASCAL_HIER1_TABLE, PASCAL_INCOMP_TABLE),
4171 P 4288 1
4172 P 4289 1
4173 P 4290 1
4174 P 4291 1
4175 P 4292 1
4176 P 4293 1 (EQUAL, PASCAL_EQL_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4177 P 4294 1 OPERATOR_INFO_ENTRY
4178 P 4295 1 (NOT_EQUAL, PASCAL_NEQ_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4179 P 4296 1 OPERATOR_INFO_ENTRY
4180 P 4297 1 (LSS_THAN, PASCAL_LSS_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4181 P 4298 1 OPERATOR_INFO_ENTRY
4182 P 4299 1 (GTR_THAN, PASCAL_GTR_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4183 P 4300 1 OPERATOR_INFO_ENTRY
4184 P 4301 1 (LSS_EQUAL, PASCAL_LEQ_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4185 P 4302 1 OPERATOR_INFO_ENTRY
4186 P 4303 1 (GTR_EQUAL, PASCAL_GEQ_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
4187 P 4304 1
4188 P 4305 1
4189 P 4306 1
4190 P 4307 1
4191 P 4308 1 OPERATOR_INFO_ENTRY
4192 P 4309 1 (AND, PASCAL_AND_TABLE, TABLEBASE, TABLEBASE),
4193 P 4310 1 OPERATOR_INFO_ENTRY
4194 P 4311 1 (OR, PASCAL_OR_TABLE, TABLEBASE, TABLEBASE),
4195 P 4312 1 OPERATOR_INFO_ENTRY
4196 P 4313 1 (NOT, PASCAL_NOT_TABLE, TABLEBASE, TABLEBASE),
4197 P 4314 1
4198 P 4315 1
4199 P 4316 1
4200 P 4317 1
4201 P 4318 1 OPERATOR_INFO_ENTRY
4202 P 4319 1 (SET_MEMBER, PASCAL_IN_TABLE, TABLEBASE, TABLEBASE),
4203 P 4320 1
4204 P 4321 1
4205 P 4322 1
4206 P 4323 1
4207 P 4324 1
4208 P 4325 1 OPERATOR_INFO_ENTRY
4209 P 4326 1 (CONVERT, TABLEBASE, PASCAL_HIERD_TABLE, PASCAL_INCOMP_TABLE),
4210 P 4327 1
4211 P 4328 1
4212 P 4329 1
4213 P 4330 1
4214 P 4331 1
4215 P 4332 1 OPERATOR_INFO_ENTRY
4216 P 4333 1 (DEPOSIT, TABLEBASE, PASCAL_HIERD_TABLE, PASCAL_INCOMP_TABLE),
4217 P 4334 1
4218 P 4335 1
4219 P 4336 1
4220 P 4337 1
4221 P 4338 1
4222 P 4339 1 OPERATOR_INFO_ENTRY
4223 P 4340 1 (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE),
4224 P 4341 1
4225 P 4342 1
4226 P 4343 1
```

(REMAINDER, PASCAL\_REM\_TABLE, PASCAL\_HIER1\_TABLE, PASCAL\_INCOMP\_TABLE),

The relational operators all use the normal HIER table and the normal incompatibility table.

OPERATOR\_INFO\_ENTRY  
(EQUAL, PASCAL\_EQL\_TABLE, PASCAL\_HIER\_TABLE, PASCAL\_INCOMP\_TABLE),

OPERATOR\_INFO\_ENTRY  
(NOT\_EQUAL, PASCAL\_NEQ\_TABLE, PASCAL\_HIER\_TABLE, PASCAL\_INCOMP\_TABLE),

OPERATOR\_INFO\_ENTRY  
(LSS\_THAN, PASCAL\_LSS\_TABLE, PASCAL\_HIER\_TABLE, PASCAL\_INCOMP\_TABLE),

OPERATOR\_INFO\_ENTRY  
(GTR\_THAN, PASCAL\_GTR\_TABLE, PASCAL\_HIER\_TABLE, PASCAL\_INCOMP\_TABLE),

OPERATOR\_INFO\_ENTRY  
(LSS\_EQUAL, PASCAL\_LEQ\_TABLE, PASCAL\_HIER\_TABLE, PASCAL\_INCOMP\_TABLE),

OPERATOR\_INFO\_ENTRY  
(GTR\_EQUAL, PASCAL\_GEQ\_TABLE, PASCAL\_HIER\_TABLE, PASCAL\_INCOMP\_TABLE),

The logical operators only operate on the type Tf. They thus need no hierarchy table and no incompatibility table.

OPERATOR\_INFO\_ENTRY  
(AND, PASCAL\_AND\_TABLE, TABLEBASE, TABLEBASE),

OPERATOR\_INFO\_ENTRY  
(OR, PASCAL\_OR\_TABLE, TABLEBASE, TABLEBASE),

OPERATOR\_INFO\_ENTRY  
(NOT, PASCAL\_NOT\_TABLE, TABLEBASE, TABLEBASE),

Set member is only allowed between the pairs given in the operator table above, with no implicit conversions.

OPERATOR\_INFO\_ENTRY  
(SET\_MEMBER, PASCAL\_IN\_TABLE, TABLEBASE, TABLEBASE),

CONVERT gets used to convert subscripts to the appropriate type. We specify the HIERD table to allow the same conversions that we allow on a DEPOSIT.

OPERATOR\_INFO\_ENTRY  
(CONVERT, TABLEBASE, PASCAL\_HIERD\_TABLE, PASCAL\_INCOMP\_TABLE),

DEPOSIT gets used in the DEPOSIT operator. The HIERD table allows, for the most part, the same combinations that the PASCAL compiler allows on assignment.

OPERATOR\_INFO\_ENTRY  
(DEPOSIT, TABLEBASE, PASCAL\_HIERD\_TABLE, PASCAL\_INCOMP\_TABLE),

The identity operator is called at the end of an EVALUATE command if we still have a Primary and we need to convert it to a Value Descriptor.

OPERATOR\_INFO\_ENTRY  
(IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE),

Built-in functions use only the tables required by the particular function. Most need not use the hierarchy table or the incompatibility

DBGEVALOP  
V04-000

6 1  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 B11ss-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32;1

Page 81  
(17)

.. 4227	P 4344	1	! table.
.. 4228	P 4345	1	!
.. 4229	P 4346	1	OPERATOR INFO ENTRY
.. 4230	P 4347	1	(SUCCESSOR, PASCAL_SUCCESSOR_TABLE, TABLEBASE, TABLEBASE),
.. 4231	P 4348	1	
.. 4232	P 4349	1	OPERATOR INFO ENTRY
.. 4233	P 4350	1	(PREDECESSOR, PASCAL_PREDECESSOR_TABLE, TABLEBASE, TABLEBASE)
.. 4234	4351	1	);
.. 4235	4352	1	

## PLI OPERATOR INFORMATION TABLES

This section contains the Operator Routine and Type tables needed to evaluate expressions in the PLI language.

## Summary of Data Types

## .Computational data types

## .Arithmetic data types

## - fix-point binary (integers)

FIXED [BINARY] [(p)]

1 &lt;= p &lt;= 7 (byte)

8 &lt;= p &lt;= 15 (word)

16 &lt;= p &lt;= 31 (longword)

default=(31)

Note: all integer constants are represented as fixed-point decimal.

## - fixed-point decimal (decimal integers and fractions)

[FIXED] DECIMAL [(p[,q])]

1 &lt;= p &lt;= 31

0 &lt;= q &lt;= p

default=(10,0)

## - floating-point (binary and decimal)

FLOAT [DECIMAL/BINARY] [(p)]

BINARY: default

1 &lt;= p &lt;= 113

p: no. of bits in mantissa

default=(24)

DECIMAL

1 &lt;= p &lt;= 34

p: no. of digits in mantissa

default=(7)

floating point values      sign bits      exponent bits      fractional bits

.F (1 &lt;= p &lt;= 7) / (1 &lt;= p &lt;= 24)

1

8

24

.D (8 &lt;= p &lt;= 15) / (25 &lt;= p &lt;= 53)

1

8

53

.G (same as above)

1

11

53

.H (16 &lt;= p &lt;= 34) / (54 &lt;= p &lt;= 113)

1

15

113

## - pictured (fixed-point data stored in character form)

PICTURE 'picture'

## .Character-string data

[CHARACTER [(n)] [VARYING]

n &lt;= 32767

default=(1)

## .Bit-string data

BIT [(n)] [ALIGNED]

n &lt;= 32767 bits

default=(1)

PLI pads the bit-string value in the direction of least significance with zeros or truncates the least significant bits from the bit-string value.

note: DECLARE ABIT BIT (10);

ABIT = '1011'B;

(high memory) 0000001101 (low memory)

'character-string'Bn is not supported.

4294 4410 1 (n is with bases other than 2. B/B1:0,1;B2:0,1,2,3  
4295 4411 1 B3:0...7;B4:0...F)  
4296 4412 1  
4297 4413 1 .Noncomputational data types  
4298 4414 1  
4299 4415 1 .Entry constants and variables  
4300 4416 1 .Label constants and variables  
4301 4417 1 .File constants and variables  
4302 4418 1 .Pointers  
4303 4419 1 .Areas and offsets  
4304 4420 1  
4305 4421 1 Aggregates  
4306 4422 1 .Arrays  
4307 4423 1 .Structures  
4308 4424 1  
4309 4425 1 Operators and Operands  
4310 4426 1 .Arithmetic (+ - / \* \*\*) Arithmetic operands  
4311 4427 1 .Relational (> < ^> ^< >= <= computational operands only  
4312 4428 1 = ^=) two operands of the same type of all  
4313 4429 1 .Logical (^ & !;) Bit-string operands  
4314 4430 1 .Concatenation (!!) Bit-string/Character-string  
4315 4431 1  
4316 4432 1 Note: / should not be used to divide two fixed-point binary operand.  
4317 4433 1  
4318 4434 1 Precedence:  
4319 4435 1 \*\* +(unary) -(unary) ^ \* / + - !! > < ^> ^< = ^= <= >= & !  
4320 4436 1  
4321 4437 1 Conversion:  
4322 4438 1 .If any operand has BINARY, the derived base is BINARY. Otherwise, the  
4323 4439 1 derived base is DECIMAL.  
4324 4440 1 .If any operand has FLOAT, the derived scale is FLOAT. Otherwise, the  
4325 4441 1 derived base is FIXED.  
4326 4442 1  
4327 4443 1 Conversion and arithmetic operation:  
4328 4444 1 fixed-point decimal integer --> fixed-point binary (with zero scale) if  
4329 4445 1 one operand is fixed-point binary.  
4330 4446 1  
4331 4447 1 note:  
4332 4448 1 ev fixbin/2 (in PLI is not allowed, we perform the operation)  
4333 4449 1 ev fixbin+2.2 (in PLI is not allowed, we perform the operation)  
4334 4450 1  
4335 4451 1 fixed-point binary/fixed-point decimal --> fixed-point float.  
4336 4452 1  
4337 4453 1 picture --> fixed decimal with precision and scale.  
4338 4454 1  
4339 4455 1 Offsets and Pointers: ptr <- ptr, offset <- offset,  
4340 4456 1 ptr <- offset, offset <- ptr (offset must have been  
4341 4457 1 declared with an area)  
4342 4458 1  
4343 4459 1 Rules for conversion of data  
4344 4460 1  
4345 4461 1 target\source : Arithmetic : Pictured : Bit String : Character String  
4346 4462 1 -----  
4347 4463 1 Arithmetic  
4348 4464 1 FIXED BINARY (fixedoverflow) Pictured Nonnegative Arithmetic  
4349 4465 1 FIXED DECIMAL src p > dst q values have FIXED BINARY constant  
4350 4466 1 FIXED (fixedoverflow) (error)

```

4351 4467 1 : FIXED BINARY (fixedoverflow) DECIMAL src value > invalid numeric
4352 4468 1 : src value > format, 2**31, characters
4353 4469 1 : dst storage followed the same -> dst type
4354 4470 1 : (fixedoverflow)
4355 4471 1 : FIXED DECIMAL (truncated) rules as rules as src p is too
4356 4472 1 : src q > dst q arith. -> arith. -> large
4357 4473 1 : (padded 0 right) arith. arith. (truncated)
4358 4474 1 : src q < dst q src q is large
4359 4475 1 :
4360 4476 1 : FLOAT (overflow) null is 0 null/spaces is 0
4361 4477 1 : src lvalue! >
4362 4478 1 : dst flt value
4363 4479 1 : (underflow)
4364 4480 1 : src lvalue!
4365 4481 1 : is too small
4366 4482 1 : -----
4367 4483 1 : Bit String takes absolute same as (padded 0) character string
4368 4484 1 : value -> arith. src n < of 0s and 1s ->
4369 4485 1 : FIXED BINARY dst n bit-string
4370 4486 1 : (31) (truncated) (error)
4371 4487 1 : (fixedoverflow) src n > any other char.
4372 4488 1 : src value > dst n
4373 4489 1 : dst value
4374 4490 1 : sign and q are
4375 4491 1 : lost during
4376 4492 1 : the conversion
4377 4493 1 : -----
4378 4494 1 : Char. String FIXED BINARY -> character bit string -> (padded trailing
4379 4495 1 : FIXED DECIMAL string char. string spaces)
4380 4496 1 : FLOAT BINARY -> represent. intmed. >
4381 4497 1 : FLOAT DECIMAL is used dst (not varying)
4382 4498 1 : (truncated)
4383 4499 1 : dst > intmed.
4384 4500 1 : -----
4385 4501 1 : Pictured src -> fixed decimal
4386 4502 1 : FIXED DECIMAL
4387 4503 1 :
4388 4504 1 :
4389 4505 1 :
4390 4506 1 : This section contains the Operator Routine and Type tables needed to
4391 4507 1 : evaluate expressions in the PASCAL language.
4392 4508 1 :
4393 4509 1 : P L I O P E R A T O R I N F O R M A T I O N T A B L E S
4394 4510 1 :
4395 4511 1 :
4396 4512 1 : This section contains the Operator Routine and Type tables needed to
4397 4513 1 : evaluate expressions in the PLI language.
4398 4514 1 :
4399 4515 1 :
4400 4516 1 :
4401 4517 1 :
4402 4518 1 : Define a Type Mapping Table for PL/I.
4403 4519 1 :
4404 4520 1 : TYPE MAPPING TABLE (PLI MAP TABLE,
4405 4521 1 : TYPE_GRAPH_EDGE (PTR, LT),
4406 4522 1 : TYPE_GRAPH_EDGE (TF, V),
4407 4523 1 : 0);

```

```
4408      4524 1
4409      4525 1
4410      4526 1 ! Define the Type Incompatibility Table for PL/I. This prevents mixing
4411      4527 1 ! D and G types in an expression or deposit.
4412      4528 1
4413      P 4529 1 TYPE_INCOMP_TABLE (PLI_INCOMP_TABLE,
4414      P 4530 1     TYPE_GRAPH_EDGE (D, G),
4415      4531 1     0);
4416      4532 1
4417      4533 1
4418      4534 1 ! Define the PL/I Specific Type Conversion Table. This allows proper
4419      4535 1 ! handling of PL/I bit-strings.
4420      4536 1
4421      P 4537 1 LANG_CVT_TABLE (PLI_CVT_TABLE,
4422      P 4538 1     LANG_CVT_ENTRY (PLI_CVT, ANY, V),
4423      P 4539 1     LANG_CVT_ENTRY (PLI_CVT, V, ANY),
4424      P 4540 1     LANG_CVT_ENTRY (PLI_CVT, ANY, VU),
4425      P 4541 1     LANG_CVT_ENTRY (PLI_CVT, VU, ANY),
4426      P 4542 1     LANG_CVT_ENTRY (PLI_CVT, ANY, T),
4427      P 4543 1     LANG_CVT_ENTRY (PLI_CVT, T, ANY),
4428      P 4544 1     LANG_CVT_ENTRY (PLI_CVT, ANY, VT),
4429      P 4545 1     LANG_CVT_ENTRY (PLI_CVT, VT, ANY),
4430      P 4546 1     LANG_CVT_ENTRY (PLI_CVT, PICT, ANY),
4431      P 4547 1     LANG_CVT_ENTRY (PLI_CVT, ANY, PICT),
4432      4548 1     0);
4433      4549 1
4434      4550 1
4435      4551 1 ! Define the Type Conversion Information Table for PLI. PL/I bit-strings
4436      4552 1 ! require PL/I specific conversions (they are stored in reverse order).
4437      4553 1
4438      P 4554 1 CONVERSION_INFO_TABLE (PLI_CVTINFO_TABLE,
4439      4555 1     CONVERSION_INFO_ENTRY (PLI_MAP_TABLE, PLI_CVT_TABLE));
4440      4556 1
4441      4557 1
4442      4558 1 ! Define the Type Hierarchy Table for PLI.
4443      4559 1 !
4444      4560 1
4445      4561 1
4446      4562 1 ! Define a Type Hierarachy Table for PLI.
4447      4563 1
4448      P 4564 1 TYPE_HIERARCHY_TABLE (PLI_HIER1_TABLE,
4449      P 4565 1
4450      P 4566 1 ! The following 4 entries are made to validate the valid operand. To
4451      P 4567 1 ! decide the target type, there is additional code to help.
4452      4568 1 !
4453      P 4569 1     TYPE_GRAPH_EDGE (T, P),
4454      P 4570 1     TYPE_GRAPH_EDGE (VF, P),
4455      P 4571 1     TYPE_GRAPH_EDGE (V, P),
4456      P 4572 1     TYPE_GRAPH_EDGE (VU, P),
4457      P 4573 1
4458      P 4574 1     TYPE_GRAPH_EDGE (PICT, P),
4459      P 4575 1     TYPE_GRAPH_EDGE (B, W),
4460      P 4576 1     TYPE_GRAPH_EDGE (W, P),
4461      P 4577 1     TYPE_GRAPH_EDGE (P, L),
4462      P 4578 1     TYPE_GRAPH_EDGE (L, F),
4463      P 4579 1     TYPE_GRAPH_EDGE (L, FIXED),
4464      P 4580 1     TYPE_GRAPH_EDGE (F, D),
```

```
4465 P 4581 1 TYPE_GRAPH_EDGE (F, G),
4466 P 4582 1 TYPE_GRAPH_EDGE (FIXED, H),
4467 P 4583 1 TYPE_GRAPH_EDGE (D, H),
4468 P 4584 1 TYPE_GRAPH_EDGE (G, H),
4469 4585 1 0);
4470 4586 1
4471 4587 1
4472 4588 1 ! Define a Type Hierarachy Table for PLI. (Relational)
4473 4589 1
4474 P 4590 1 TYPE_HIERARCHY_TABLE (PLI_HIER2_TABLE,
4475 P 4591 1 TYPE_GRAPH_EDGE (VT, P),
4476 P 4592 1 TYPE_GRAPH_EDGE (VT, T),
4477 P 4593 1 TYPE_GRAPH_EDGE (T, P),
4478 P 4594 1 TYPE_GRAPH_EDGE (VU, P),
4479 P 4595 1 TYPE_GRAPH_EDGE (VU, T),
4480 P 4596 1 TYPE_GRAPH_EDGE (VU, VT),
4481 P 4597 1 TYPE_GRAPH_EDGE (VU, V),
4482 P 4598 1 TYPE_GRAPH_EDGE (V, P),
4483 P 4599 1 TYPE_GRAPH_EDGE (V, T),
4484 P 4600 1 TYPE_GRAPH_EDGE (V, VT),
4485 P 4601 1
4486 P 4602 1 TYPE_GRAPH_EDGE (PICT, P),
4487 P 4603 1 TYPE_GRAPH_EDGE (B, W),
4488 P 4604 1 TYPE_GRAPH_EDGE (W, P),
4489 P 4605 1 TYPE_GRAPH_EDGE (P, L),
4490 P 4606 1 TYPE_GRAPH_EDGE (L, F),
4491 P 4607 1 TYPE_GRAPH_EDGE (L, FIXED),
4492 P 4608 1 TYPE_GRAPH_EDGE (F, D),
4493 P 4609 1 TYPE_GRAPH_EDGE (F, G),
4494 P 4610 1 TYPE_GRAPH_EDGE (FIXED, H),
4495 P 4611 1 TYPE_GRAPH_EDGE (D, H),
4496 P 4612 1 TYPE_GRAPH_EDGE (G, H),
4497 4613 1 0);
4498 4614 1
4499 4615 1
4500 4616 1 ! Define a Type Hierarachy Table for PLI. (Logical)
4501 4617 1
4502 P 4618 1 TYPE_HIERARCHY_TABLE (PLI_HIER3_TABLE,
4503 P 4619 1 TYPE_GRAPH_EDGE (B, V),
4504 P 4620 1 TYPE_GRAPH_EDGE (W, V),
4505 P 4621 1 TYPE_GRAPH_EDGE (L, V),
4506 P 4622 1 TYPE_GRAPH_EDGE (P, V),
4507 P 4623 1 TYPE_GRAPH_EDGE (F, V),
4508 P 4624 1 TYPE_GRAPH_EDGE (D, V),
4509 P 4625 1 TYPE_GRAPH_EDGE (G, V),
4510 P 4626 1 TYPE_GRAPH_EDGE (H, V),
4511 P 4627 1 TYPE_GRAPH_EDGE (PICT, V),
4512 P 4628 1 TYPE_GRAPH_EDGE (T, V),
4513 P 4629 1 TYPE_GRAPH_EDGE (VT, V),
4514 P 4630 1 TYPE_GRAPH_EDGE (VU, V),
4515 4631 1 0);
4516 4632 1
4517 4633 1
4518 4634 1 ! Define a Type Hierarachy Table for PLI. (Concatination)
4519 4635 1
4520 P 4636 1 TYPE_HIERARCHY_TABLE (PLI_HIER4_TABLE,
4521 P 4637 1 TYPE_GRAPH_EDGE (B, T),
```

```
.. 4522 P 4638 1 TYPE_GRAPH_EDGE (W, T).
.. 4523 P 4639 1 TYPE_GRAPH_EDGE (L, T).
.. 4524 P 4640 1 TYPE_GRAPH_EDGE (P, T).
.. 4525 P 4641 1 TYPE_GRAPH_EDGE (F, T).
.. 4526 P 4642 1 TYPE_GRAPH_EDGE (D, T).
.. 4527 P 4643 1 TYPE_GRAPH_EDGE (G, T).
.. 4528 P 4644 1 TYPE_GRAPH_EDGE (H, T).
.. 4529 P 4645 1 TYPE_GRAPH_EDGE (PICT, T).
.. 4530 P 4646 1 TYPE_GRAPH_EDGE (VT, T).
.. 4531 P 4647 1 TYPE_GRAPH_EDGE (VU, V).
.. 4532 P 4648 1 TYPE_GRAPH_EDGE (V, T).
.. 4533 4649 1 0);
.. 4534 4650 1
.. 4535 4651 1
.. 4536 4652 1 ! Define a Type Hierarachy Table for PLI. (Unary +/-)
.. 4537 4653 1
.. 4538 P 4654 1 TYPE_HIERARCHY_TABLE (PLI_HIERS_TABLE,
.. 4539 P 4655 1 TYPE_GRAPH_EDGE (T, PT).
.. 4540 P 4656 1 TYPE_GRAPH_EDGE (VT, P).
.. 4541 P 4657 1 TYPE_GRAPH_EDGE (V, L).
.. 4542 P 4658 1 TYPE_GRAPH_EDGE (VU, L).
.. 4543 P 4659 1 TYPE_GRAPH_EDGE (PICT, P).
.. 4544 P 4660 1 TYPE_GRAPH_EDGE (B, W).
.. 4545 P 4661 1 TYPE_GRAPH_EDGE (W, P).
.. 4546 P 4662 1 TYPE_GRAPH_EDGE (P, L).
.. 4547 P 4663 1 TYPE_GRAPH_EDGE (L, F).
.. 4548 P 4664 1 TYPE_GRAPH_EDGE (L, FIXED).
.. 4549 P 4665 1 TYPE_GRAPH_EDGE (F, D).
.. 4550 P 4666 1 TYPE_GRAPH_EDGE (F, G).
.. 4551 P 4667 1 TYPE_GRAPH_EDGE (FIXED, H).
.. 4552 P 4668 1 TYPE_GRAPH_EDGE (D, H).
.. 4553 P 4669 1 TYPE_GRAPH_EDGE (G, H).
.. 4554 4670 1 0);
.. 4555 4671 1
.. 4556 4672 1
.. 4557 4673 1 ! For PLI DEPOSIT.
.. 4558 4674 1
.. 4559 P 4675 1 TYPE_HIERARCHY_TABLE (PLI_HIERSD_TABLE,
.. 4560 P 4676 1 TYPE_GRAPH_EDGE (B, W).
.. 4561 P 4677 1 TYPE_GRAPH_EDGE (W, L).
.. 4562 P 4678 1 TYPE_GRAPH_EDGE (L, P).
.. 4563 P 4679 1 TYPE_GRAPH_EDGE (P, FIXED).
.. 4564 P 4680 1 TYPE_GRAPH_EDGE (FIXED, F).
.. 4565 P 4681 1 TYPE_GRAPH_EDGE (F, D).
.. 4566 P 4682 1 TYPE_GRAPH_EDGE (D, G).
.. 4567 P 4683 1 TYPE_GRAPH_EDGE (G, H).
.. 4568 P 4684 1 TYPE_GRAPH_EDGE (H, V).
.. 4569 P 4685 1 TYPE_GRAPH_EDGE (V, VU).
.. 4570 P 4686 1 TYPE_GRAPH_EDGE (VU, T).
.. 4571 P 4687 1 TYPE_GRAPH_EDGE (T, VT).
.. 4572 P 4688 1 TYPE_GRAPH_EDGE (VT, PICT).
.. 4573 P 4689 1 TYPE_GRAPH_EDGE (PICT, B).
.. 4574 4690 1 0);
.. 4575 4691 1
.. 4576 4692 1
.. 4577 4693 1 ! Most of the arithmetic routines operate on two arguments of the same type.
.. 4578 4694 1 ! That type may be B, W, L, F, D, G, H, P, so we provide all of those case
```

```
4579      4695 1 | indices.
4580      4696 1 |
4581      4697 1 | Define the Operator Routine Table for PLI addition.
4582      4698 1 |
4583      P 4699 1 | OPERATOR ROUTINE TABLE (PLI_ADD TABLE,
4584      P 4700 1 |     OPERATOR_ROUTINE (B, B, B, ADD_B_B),
4585      P 4701 1 |     OPERATOR_ROUTINE (W, W, W, ADD_W_W),
4586      P 4702 1 |     OPERATOR_ROUTINE (L, L, L, ADD_L_L),
4587      P 4703 1 |     OPERATOR_ROUTINE (F, F, F, ADD_F_F),
4588      P 4704 1 |     OPERATOR_ROUTINE (D, D, D, ADD_D_D),
4589      P 4705 1 |     OPERATOR_ROUTINE (G, G, G, ADD_G_G),
4590      P 4706 1 |     OPERATOR_ROUTINE (H, H, H, ADD_H_H),
4591      P 4707 1 |     OPERATOR_ROUTINE (P, P, P, ADD_P_P),
4592      4708 1 |     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, ADD_FIXED_FIXED));
4593      4709 1 |
4594      4710 1 |
4595      4711 1 | Define the Operator Routine Table for PLI subtraction.
4596      4712 1 |
4597      P 4713 1 | OPERATOR ROUTINE TABLE (PLI_SUB TABLE,
4598      P 4714 1 |     OPERATOR_ROUTINE (B, B, B, SUB_B_B),
4599      P 4715 1 |     OPERATOR_ROUTINE (W, W, W, SUB_W_W),
4600      P 4716 1 |     OPERATOR_ROUTINE (L, L, L, SUB_L_L),
4601      P 4717 1 |     OPERATOR_ROUTINE (F, F, F, SUB_F_F),
4602      P 4718 1 |     OPERATOR_ROUTINE (D, D, D, SUB_D_D),
4603      P 4719 1 |     OPERATOR_ROUTINE (G, G, G, SUB_G_G),
4604      P 4720 1 |     OPERATOR_ROUTINE (H, H, H, SUB_H_H),
4605      P 4721 1 |     OPERATOR_ROUTINE (P, P, P, SUB_P_P),
4606      4722 1 |     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, SUB_FIXED_FIXED));
4607      4723 1 |
4608      4724 1 |
4609      4725 1 | Define the Operator Routine Table for PLI multiplication.
4610      4726 1 |
4611      P 4727 1 | OPERATOR ROUTINE TABLE (PLI_MUL TABLE,
4612      P 4728 1 |     OPERATOR_ROUTINE (B, B, B, MUL_B_B),
4613      P 4729 1 |     OPERATOR_ROUTINE (W, W, W, MUL_W_W),
4614      P 4730 1 |     OPERATOR_ROUTINE (L, L, L, MUL_L_L),
4615      P 4731 1 |     OPERATOR_ROUTINE (F, F, F, MUL_F_F),
4616      P 4732 1 |     OPERATOR_ROUTINE (D, D, D, MUL_D_D),
4617      P 4733 1 |     OPERATOR_ROUTINE (G, G, G, MUL_G_G),
4618      P 4734 1 |     OPERATOR_ROUTINE (H, H, H, MUL_H_H),
4619      P 4735 1 |     OPERATOR_ROUTINE (P, P, P, MUL_P_P),
4620      4736 1 |     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, MUL_FIXED_FIXED));
4621      4737 1 |
4622      4738 1 |
4623      4739 1 | Define the Operator Routine Table for PLI division.
4624      4740 1 |
4625      P 4741 1 | OPERATOR ROUTINE TABLE (PLI_DIV TABLE,
4626      P 4742 1 |     OPERATOR_ROUTINE (B, B, B, DIV_B_B),
4627      P 4743 1 |     OPERATOR_ROUTINE (W, W, W, DIV_W_W),
4628      P 4744 1 |     OPERATOR_ROUTINE (L, L, L, DIV_L_L),
4629      P 4745 1 |     OPERATOR_ROUTINE (F, F, F, DIV_F_F),
4630      P 4746 1 |     OPERATOR_ROUTINE (D, D, D, DIV_D_D),
4631      P 4747 1 |     OPERATOR_ROUTINE (G, G, G, DIV_G_G),
4632      P 4748 1 |     OPERATOR_ROUTINE (H, H, H, DIV_H_H),
4633      P 4749 1 |     OPERATOR_ROUTINE (P, P, P, DIV_P_P),
4634      4750 1 |     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, DIV_FIXED_FIXED));
4635      4751 1 |
```

```
4636 4752 1
4637 4753 1 !! Define the Operator Routine Table for PLI unary plus.
4638 4754 1
4639 P 4755 1 OPERATOR_ROUTINE_TABLE (PLI_UNARY_PLUS_TABLE,
4640 P 4756 1
4641 P 4757 1 !! The following are not language dependent types. This is needed for DEBUG
4642 P 4758 1 !! types. For example, DEP/QUAD L= +1.
4643 P 4759 1
4644 P 4760 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
4645 P 4761 1 OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
4646 P 4762 1
4647 P 4763 1 OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
4648 P 4764 1 OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
4649 P 4765 1 OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
4650 P 4766 1 OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
4651 P 4767 1 OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
4652 P 4768 1 OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
4653 P 4769 1 OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
4654 P 4770 1 OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
4655 4771 1 OPERATOR_ROUTINE (FIXED, FIXED, FIXED, UNARY_PLUS_FIXED));
4656 4772 1
4657 4773 1
4658 4774 1 !! Define the Operator Routine Table for PLI unary minus.
4659 4775 1
4660 P 4776 1 OPERATOR_ROUTINE_TABLE (PLI_UNARY_MINUS_TABLE,
4661 P 4777 1
4662 P 4778 1 !! The following are not language dependent types. This is needed for DEBUG
4663 P 4779 1 !! types. For example, DEP/QUAD L= +1.
4664 P 4780 1
4665 P 4781 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
4666 P 4782 1 OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
4667 P 4783 1
4668 P 4784 1 OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
4669 P 4785 1 OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
4670 P 4786 1 OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
4671 P 4787 1 OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
4672 P 4788 1 OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
4673 P 4789 1 OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
4674 P 4790 1 OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
4675 P 4791 1 OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
4676 4792 1 OPERATOR_ROUTINE (FIXED, FIXED, FIXED, UNARY_MINUS_FIXED));
4677 4793 1
4678 4794 1
4679 4795 1 !! Define the Operator Routine Table for PLI exponentiation.
4680 4796 1 !! Exponentiation has some mixed forms. For example, if you raise a
4681 4797 1 !! floating number to an integer power, you do not necessarily
4682 4798 1 !! want to first convert the int to float. Instead, a special
4683 4799 1 !! routine indices such as POWER_F_L are provided to do the right thing here.
4684 4800 1
4685 P 4801 1 OPERATOR_ROUTINE_TABLE (PLI_POWER_TABLE,
4686 P 4802 1 OPERATOR_ROUTINE (F, L, F, POWER_F_L),
4687 P 4803 1 OPERATOR_ROUTINE (D, L, D, POWER_D_L),
4688 P 4804 1 OPERATOR_ROUTINE (G, L, G, POWER_G_L),
4689 P 4805 1 OPERATOR_ROUTINE (H, L, H, POWER_H_L),
4690 P 4806 1 OPERATOR_ROUTINE (F, F, F, POWER_F_F),
4691 P 4807 1 OPERATOR_ROUTINE (D, F, D, POWER_D_F),
4692 P 4808 1 OPERATOR_ROUTINE (F, D, D, POWER_F_D),
```

```
4693 P 4809 1 OPERATOR_ROUTINE (D, D, D, POWER_D_D),
4694 P 4810 1 OPERATOR_ROUTINE (G, G, G, POWER_G_G),
4695 4811 1 OPERATOR_ROUTINE (H, H, H, POWER_H_H));
4696 4812 1
4697 4813 1
4698 4814 1 ! Define the Operator Routine Table for PLI concatenation.
4699 4815 1 ! Concatenate can only be done on character or bit strings.
4700 4816 1
4701 P 4817 1 OPERATOR_ROUTINE_TABLE (PLI_CONCAT_TABLE,
4702 P 4818 1 OPERATOR_ROUTINE (T, T, T, CONCAT_T_T),
4703 4819 1 OPERATOR_ROUTINE (V, V, V, CONCAT_VF_VF));
4704 4820 1
4705 4821 1
4706 4822 1 ! In the tables for the comparison operators, we allow strings to be
4707 4823 1 ! compared, and also all the numeric types.
4708 4824 1
4709 4825 1 ! Define the Operator Routine Table for PLI equal.
4710 4826 1
4711 P 4827 1 OPERATOR_ROUTINE_TABLE (PLI_EQL_TABLE,
4712 P 4828 1 OPERATOR_ROUTINE (B, B, TF, EQL_B_B),
4713 P 4829 1 OPERATOR_ROUTINE (W, W, TF, EQL_W_W),
4714 P 4830 1 OPERATOR_ROUTINE (L, L, TF, EQL_L_L),
4715 P 4831 1 OPERATOR_ROUTINE (F, F, TF, EQL_F_F),
4716 P 4832 1 OPERATOR_ROUTINE (D, D, TF, EQL_D_D),
4717 P 4833 1 OPERATOR_ROUTINE (G, G, TF, EQL_G_G),
4718 P 4834 1 OPERATOR_ROUTINE (H, H, TF, EQL_H_H),
4719 P 4835 1 OPERATOR_ROUTINE (P, P, TF, EQL_P_P),
4720 P 4836 1 OPERATOR_ROUTINE (FIXED, FIXED, TF, EQL_FIXED_FIXED),
4721 P 4837 1 OPERATOR_ROUTINE (T, T, TF, EQL_T_T),
4722 P 4838 1 OPERATOR_ROUTINE (VF, VF, TF, EQL_VF_VF),
4723 P 4839 1 OPERATOR_ROUTINE (V, V, TF, EQL_VF_VF),
4724 P 4840 1 OPERATOR_ROUTINE (VU, VU, TF, EQL_VF_VF),
4725 4841 1
4726 P 4842 1 ! This one should have non-computational data item as well.
4727 P 4843 1 !
4728 P 4844 1 ! entry, label, file, pointer (which we map into L), area and offsets
4729 P 4845 1 !
4730 4846 1 );
4731 4847 1
4732 4848 1
4733 4849 1 ! Define the Operator Routine Table for PLI not equal.
4734 4850 1
4735 P 4851 1 OPERATOR_ROUTINE_TABLE (PLI_NEQ_TABLE,
4736 P 4852 1 OPERATOR_ROUTINE (B, B, TF, NEQ_B_B),
4737 P 4853 1 OPERATOR_ROUTINE (W, W, TF, NEQ_W_W),
4738 P 4854 1 OPERATOR_ROUTINE (L, L, TF, NEQ_L_L),
4739 P 4855 1 OPERATOR_ROUTINE (F, F, TF, NEQ_F_F),
4740 P 4856 1 OPERATOR_ROUTINE (D, D, TF, NEQ_D_D),
4741 P 4857 1 OPERATOR_ROUTINE (G, G, TF, NEQ_G_G),
4742 P 4858 1 OPERATOR_ROUTINE (H, H, TF, NEQ_H_H),
4743 P 4859 1 OPERATOR_ROUTINE (P, P, TF, NEQ_P_P),
4744 P 4860 1 OPERATOR_ROUTINE (FIXED, FIXED, TF, NEQ_FIXED_FIXED),
4745 P 4861 1 OPERATOR_ROUTINE (T, T, TF, NEQ_T_T),
4746 P 4862 1 OPERATOR_ROUTINE (VF, VF, TF, NEQ_VF_VF),
4747 P 4863 1 OPERATOR_ROUTINE (V, V, TF, NEQ_VF_VF),
4748 P 4864 1 OPERATOR_ROUTINE (VU, VU, TF, NEQ_VF_VF),
4749 P 4865 1
```

```
4750 P 4866 1
4751 P 4867 1
4752 P 4868 1
4753 P 4869 1
4754 P 4870 1
4755 4871 1
4756 4872 1
4757 4873 1
4758 4874 1
4759 4875 1
4760 P 4876 1
4761 P 4877 1
4762 P 4878 1
4763 P 4879 1
4764 P 4880 1
4765 P 4881 1
4766 P 4882 1
4767 P 4883 1
4768 P 4884 1
4769 P 4885 1
4770 P 4886 1
4771 P 4887 1
4772 P 4888 1
4773 4889 1
4774 4890 1
4775 4891 1
4776 4892 1
4777 4893 1
4778 P 4894 1
4779 P 4895 1
4780 P 4896 1
4781 P 4897 1
4782 P 4898 1
4783 P 4899 1
4784 P 4900 1
4785 P 4901 1
4786 P 4902 1
4787 P 4903 1
4788 P 4904 1
4789 P 4905 1
4790 P 4906 1
4791 4907 1
4792 4908 1
4793 4909 1
4794 4910 1
4795 4911 1
4796 P 4912 1
4797 P 4913 1
4798 P 4914 1
4799 P 4915 1
4800 P 4916 1
4801 P 4917 1
4802 P 4918 1
4803 P 4919 1
4804 P 4920 1
4805 P 4921 1
4806 P 4922 1

! This one should have non-computational data item as well.
! entry, label, file, pointer (which we map into L), area and offsets
! );

! Define the Operator Routine Table for PLI greater than.
OPERATOR_ROUTINE_TABLE (PLI_GTR_TABLE,
  OPERATOR_ROUTINE (B, B, TF, GTR_B_B),
  OPERATOR_ROUTINE (W, W, TF, GTR_W_W),
  OPERATOR_ROUTINE (L, L, TF, GTR_L_L),
  OPERATOR_ROUTINE (F, F, TF, GTR_F_F),
  OPERATOR_ROUTINE (D, D, TF, GTR_D_D),
  OPERATOR_ROUTINE (G, G, TF, GTR_G_G),
  OPERATOR_ROUTINE (H, H, TF, GTR_H_H),
  OPERATOR_ROUTINE (P, P, TF, GTR_P_P),
  OPERATOR_ROUTINE (FIXED, FIXED, TF, GTR_FIXED_FIXED),
  OPERATOR_ROUTINE (T, T, TF, GTR_T_T),
  OPERATOR_ROUTINE (VT, VT, TF, GTR_VT_VT),
  OPERATOR_ROUTINE (VU, VU, TF, GTR_VF_VF),
  OPERATOR_ROUTINE (V, V, TF, GTR_VF_VF));

! Define the Operator Routine Table for PLI greater than or equal to.
OPERATOR_ROUTINE_TABLE (PLI_GEQ_TABLE,
  OPERATOR_ROUTINE (B, B, TF, GEQ_B_B),
  OPERATOR_ROUTINE (W, W, TF, GEQ_W_W),
  OPERATOR_ROUTINE (L, L, TF, GEQ_L_L),
  OPERATOR_ROUTINE (F, F, TF, GEQ_F_F),
  OPERATOR_ROUTINE (D, D, TF, GEQ_D_D),
  OPERATOR_ROUTINE (G, G, TF, GEQ_G_G),
  OPERATOR_ROUTINE (H, H, TF, GEQ_H_H),
  OPERATOR_ROUTINE (P, P, TF, GEQ_P_P),
  OPERATOR_ROUTINE (FIXED, FIXED, TF, GEQ_FIXED_FIXED),
  OPERATOR_ROUTINE (T, T, TF, GEQ_T_T),
  OPERATOR_ROUTINE (VT, VT, TF, GEQ_VT_VT),
  OPERATOR_ROUTINE (VU, VU, TF, GEQ_VF_VF),
  OPERATOR_ROUTINE (V, V, TF, GEQ_VF_VF));

! Define the Operator Routine Table for PLI less than.
OPERATOR_ROUTINE_TABLE (PLI_LSS_TABLE,
  OPERATOR_ROUTINE (B, B, TF, LSS_B_B),
  OPERATOR_ROUTINE (W, W, TF, LSS_W_W),
  OPERATOR_ROUTINE (L, L, TF, LSS_L_L),
  OPERATOR_ROUTINE (F, F, TF, LSS_F_F),
  OPERATOR_ROUTINE (D, D, TF, LSS_D_D),
  OPERATOR_ROUTINE (G, G, TF, LSS_G_G),
  OPERATOR_ROUTINE (H, H, TF, LSS_H_H),
  OPERATOR_ROUTINE (P, P, TF, LSS_P_P),
  OPERATOR_ROUTINE (FIXED, FIXED, TF, LSS_FIXED_FIXED),
  OPERATOR_ROUTINE (T, T, TF, LSS_T_T),
```

```
4807 P 4923 1 OPERATOR_ROUTINE (VT, VT, TF, LSS_VT_VT);
4808 P 4924 1 OPERATOR_ROUTINE (VU, VU, TF, LSS_TF_TF);
4809 4925 1 OPERATOR_ROUTINE (V, V, TF, LSS_TF_TF));
4810 4926 1
4811 4927 1
4812 4928 1 ! Define the Operator Routine Table for PLI less than or equal to.
4813 4929 1
4814 P 4930 1 OPERATOR_ROUTINE_TABLE (PLI_LEQ_TABLE,
4815 P 4931 1 OPERATOR_ROUTINE (B, B, TF, LEQ_B_B),
4816 P 4932 1 OPERATOR_ROUTINE (W, W, TF, LEQ_W_W),
4817 P 4933 1 OPERATOR_ROUTINE (L, L, TF, LEQ_L_L),
4818 P 4934 1 OPERATOR_ROUTINE (F, F, TF, LEQ_F_F),
4819 P 4935 1 OPERATOR_ROUTINE (D, D, TF, LEQ_D_D),
4820 P 4936 1 OPERATOR_ROUTINE (G, G, TF, LEQ_G_G),
4821 P 4937 1 OPERATOR_ROUTINE (H, H, TF, LEQ_H_H),
4822 P 4938 1 OPERATOR_ROUTINE (P, P, TF, LEQ_P_P),
4823 P 4939 1 OPERATOR_ROUTINE (FIXED, FIXED, TF, LEQ_FIXED_FIXED),
4824 P 4940 1 OPERATOR_ROUTINE (T, T, TF, LEQ_T_T),
4825 P 4941 1 OPERATOR_ROUTINE (VT, VT, TF, LEQ_VT_VT),
4826 P 4942 1 OPERATOR_ROUTINE (VU, VU, TF, LEQ_TF_TF),
4827 4943 1 OPERATOR_ROUTINE (V, V, TF, LEQ_TF_TF));
4828 4944 1
4829 4945 1
4830 4946 1 ! The logical operators AND, OR, and NOT can be applied only to
4831 4947 1 bit-string data types.
4832 4948 1
4833 4949 1 ! Define the Operator Routine Table for PLI NOT.
4834 4950 1
4835 P 4951 1 OPERATOR_ROUTINE_TABLE (PLI_BIT_NOT_TABLE,
4836 4952 1 OPERATOR_ROUTINE (V, V, V, BIT_NOT_TF));
4837 4953 1
4838 4954 1
4839 4955 1 ! Define the Operator Routine Table for PLI AND.
4840 4956 1
4841 P 4957 1 OPERATOR_ROUTINE_TABLE (PLI_BIT_AND_TABLE,
4842 4958 1 OPERATOR_ROUTINE (V, V, V, BIT_AND_TF));
4843 4959 1
4844 4960 1
4845 4961 1 ! Define the Operator Routine Table for PLI OR.
4846 4962 1
4847 P 4963 1 OPERATOR_ROUTINE_TABLE (PLI_BIT_OR_TABLE,
4848 4964 1 OPERATOR_ROUTINE (V, V, V, BIT_OR_TF));
4849 4965 1
4850 4966 1
4851 4967 1 ! Define the Operator Information Table for PLI.
4852 4968 1
4853 P 4969 1 OPERATOR_INFO_TABLE (PLI_OPINFO_TABLE,
4854 P 4970 1
4855 P 4971 1
4856 P 4972 1 ! The following are arithmetic tables that accept all numeric data types.
4857 P 4973 1 ! They thus need to specify an incompatibility table.
4858 P 4974 1
4859 P 4975 1 OPERATOR_INFO_ENTRY (ADD, PLI_ADD_TABLE, PLI_HIER1_TABLE,
4860 P 4976 1 PLI_INCOMP_TABLE),
4861 P 4977 1 OPERATOR_INFO_ENTRY (SUBTRACT, PLI_SUB_TABLE, PLI_HIER1_TABLE,
4862 P 4978 1 PLI_INCOMP_TABLE),
4863 P 4979 1 OPERATOR_INFO_ENTRY (MULTIPLY, PLI_MUL_TABLE, PLI_HIER1_TABLE,
```

```
4864 P 4980 1 PLI_INCOMP_TABLE),
4865 P 4981 1 OPERATOR_INFO_ENTRY (DIVIDE, PLI_DIV_TABLE, PLI_HIER1_TABLE,
4866 P 4982 1 PLI_INCOMP_TABLE),
4867 P 4983 1 OPERATOR_INFO_ENTRY (UNARY_PLUS, PLI_UNARY_PLUS_TABLE,
4868 P 4984 1 PLI_HIER5_TABLE, PLI_INCOMP_TABLE),
4869 P 4985 1 OPERATOR_INFO_ENTRY (UNARY_MINUS, PLI_UNARY_MINUS_TABLE,
4870 P 4986 1 PLI_HIER5_TABLE, PLI_INCOMP_TABLE),
4871 P 4987 1 OPERATOR_INFO_ENTRY (POWER_OF, PLI_POWER_TABLE, PLI_HIER1_TABLE,
4872 P 4988 1 PLI_INCOMP_TABLE),
4873 P 4989 1
4874 P 4990 1
4875 P 4991 1 ! The relationals accept all numeric and bit-string types.
4876 P 4992 1 ! There is no incompatibility table.
4877 P 4993 1
4878 P 4994 1 OPERATOR_INFO_ENTRY (EQUAL, PLI_EQL_TABLE, PLI_HIER2_TABLE,
4879 P 4995 1 PLI_INCOMP_TABLE),
4880 P 4996 1 OPERATOR_INFO_ENTRY (NOT_EQUAL, PLI_NEQ_TABLE, PLI_HIER2_TABLE,
4881 P 4997 1 PLI_INCOMP_TABLE),
4882 P 4998 1 OPERATOR_INFO_ENTRY (GTR_THAN, PLI_GTR_TABLE, PLI_HIER2_TABLE,
4883 P 4999 1 PLI_INCOMP_TABLE),
4884 P 5000 1 OPERATOR_INFO_ENTRY (GTR_EQUAL, PLI_GEQ_TABLE, PLI_HIER2_TABLE,
4885 P 5001 1 PLI_INCOMP_TABLE),
4886 P 5002 1 OPERATOR_INFO_ENTRY (LSS_THAN, PLI_LSS_TABLE, PLI_HIER2_TABLE,
4887 P 5003 1 PLI_INCOMP_TABLE),
4888 P 5004 1 OPERATOR_INFO_ENTRY (LSS_EQUAL, PLI_LEQ_TABLE, PLI_HIER2_TABLE,
4889 P 5005 1 PLI_INCOMP_TABLE),
4890 P 5006 1
4891 P 5007 1
4892 P 5008 1 ! The logical operators accept only bit-string quantities so they do
4893 P 5009 1 ! not need a hierarchy table. They also do not need an incompatibility
4894 P 5010 1 ! table.
4895 P 5011 1
4896 P 5012 1 OPERATOR_INFO_ENTRY (BIT_NOT, PLI_BIT_NOT_TABLE, PLI_HIER3_TABLE,
4897 P 5013 1 PLI_INCOMP_TABLE),
4898 P 5014 1 OPERATOR_INFO_ENTRY (BIT_AND, PLI_BIT_AND_TABLE, PLI_HIER3_TABLE,
4899 P 5015 1 PLI_INCOMP_TABLE),
4900 P 5016 1 OPERATOR_INFO_ENTRY (BIT_OR, PLI_BIT_OR_TABLE, PLI_HIER3_TABLE,
4901 P 5017 1 PLI_INCOMP_TABLE),
4902 P 5018 1
4903 P 5019 1
4904 P 5020 1 ! Concatenation allows only character or bit string operands, so it does
4905 P 5021 1 ! not need a hierarchy table. The incompatibility table is not required,
4906 P 5022 1 ! either.
4907 P 5023 1
4908 P 5024 1 OPERATOR_INFO_ENTRY (CONCATENATE, PLI_CONCAT_TABLE, PLI_HIER4_TABLE,
4909 P 5025 1 PLI_INCOMP_TABLE),
4910 P 5026 1
4911 P 5027 1 OPERATOR_INFO_ENTRY (CONVERT, TABLEBASE, PLI_HIERD_TABLE,
4912 P 5028 1 TABLEBASE),
4913 P 5029 1
4914 P 5030 1
4915 P 5031 1 ! The DEPOSIT operator gets called on the DEPOSIT command. It has
4916 P 5032 1 ! its own hierarchy table which allows any numeric type to be
4917 P 5033 1 ! converted to any other numeric type. The incompatibility table,
4918 P 5034 1 ! however, still prevents depositing D types into G types and
4919 P 5035 1 ! vice versa.
4920 P 5036 1 !
```

DBGEVALOP  
V04-000

6 2  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32;1

Page 94  
(18)

```
: 4921 P 5037 1
: 4922 P 5038 1
: 4923 P 5039 1
: 4924 P 5040 1
: 4925 P 5041 1
: 4926 P 5042 1
: 4927 P 5043 1
: 4928 P 5044 1
: 4929 P 5045 1
: 4930 P 5046 1
: 4931 P 5047 1
: 4932 P 5048 1
: 4933 P 5049 1
: 4934 P 5050 1
```

OPERATOR\_INFO\_ENTRY (DEPOSIT, TABLEBASE, PLI\_HIERD\_TABLE,  
PLI\_INCOMP\_TABLE),

! The IDENTITY operator gets called at the end of an EVALUATE command  
! to apply the PRIM\_TO\_VAL routine and then apply the appropriate  
! type mappings. This will ensure that EV BU will print as a signed integer,  
! for example.

! The identity operator does not require any tables.

OPERATOR\_INFO\_ENTRY (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE)  
);

```

4936 5051 1
4937 5052 1
4938 5053 1
4939 5054 1
4940 5055 1
4941 5056 1
4942 5057 1
4943 5058 1
4944 5059 1
4945 5060 1
4946 5061 1
4947 5062 1
4948 5063 1
4949 5064 1
4950 5065 1
4951 5066 1
4952 5067 1
4953 5068 1
4954 5069 1
4955 5070 1
4956 5071 1
4957 5072 1
4958 5073 1
4959 5074 1
4960 5075 1
4961 5076 1
4962 5077 1
4963 5078 1
4964 5079 1
4965 5080 1
4966 5081 1
4967 5082 1
4968 5083 1
4969 5084 1
4970 5085 1
4971 5086 1
4972 5087 1
4973 5088 1
4974 5089 1
4975 5090 1
4976 5091 1
4977 5092 1
4978 5093 1
4979 5094 1
4980 5095 1
4981 5096 1
4982 5097 1
4983 5098 1
4984 5099 1
4985 5100 1
4986 5101 1
4987 P 5102 1
4988 5103 1
4989 5104 1
4990 5105 1
4991 5106 1
4992 P 5107 1

      R P G   O P E R A T O R   I N F O R M A T I O N   T A B L E S

This section contains the Operator Routine and Type tables needed to
evaluate expressions in the RPG language.

The following summarizes the information in the RPG-11 manual,
RPGV1DPLN by Treggiari.

There is further documentation within the tables below, describing
exactly how we translate this into the DEBUG tables.

RPG Data Types:

    Trailing overpunched numeric (scaled)
    Packed numeric                (scaled)
    word                          (scaled)
    long-word                     (scaled)
    character

    1-dimensional array of the above types
    table data type
    special registers which begin with an asterisk

Expressions:
    .Arithmetic
      operator: + - * / unary+ unary-
      operand: numeric literal/identifier

    .Conditional (TRUE, FALSE)
      operator: =, NOT =, >, NOT >, <, NOT <
      operand: identifier, literal, or AE

Type Conversion:

    W -----+
    L -----+
    NRO-----+----> P

    If one of the operand is Scaled Descriptor or Packed decimal
    data type the operation is always done in Packed form. The
    operation even includes the deposit.

Define the Type Conversion Information Table for RPG. No rounding
takes place in RPG.

CONVERSION INFO TABLE (RPG CVTINFO TABLE,
CONVERSION_INFO_ENTRY (TABLEBASE, TABLEBASE));

Define Type Hierarchy Table for RPG.

TYPE_HIERARCHY_TABLE (RPG_HIER_TABLE,
```

```
4993 P 5108 1 TYPE_GRAPH_EDGE (W, P),
4994 P 5109 1 TYPE_GRAPH_EDGE (L, P),
4995 P 5110 1 TYPE_GRAPH_EDGE (NR0, P),
4996 5111 1 0);
4997 5112 1
4998 5113 1 ! Define the Type Hierarchy Table for RPG deposit.
4999 5114 1
5000 P 5115 1 TYPE_HIERARCHY_TABLE (RPG_HIERD_TABLE,
5001 P 5116 1 TYPE_GRAPH_EDGE (W, L),
5002 P 5117 1 TYPE_GRAPH_EDGE (L, NR0),
5003 P 5118 1 TYPE_GRAPH_EDGE (NR0, P),
5004 P 5119 1 TYPE_GRAPH_EDGE (P, W),
5005 5120 1 0);
5006 5121 1
5007 5122 1 ! Define the Operator Routine Table for RPG addition.
5008 5123 1
5009 P 5124 1 OPERATOR_ROUTINE_TABLE (RPG_ADD_TABLE,
5010 5125 1 OPERATOR_ROUTINE (P, P, P, ADD_P_P));
5011 5126 1
5012 5127 1 ! Define the Operator Routine Table for RPG subtraction.
5013 5128 1
5014 P 5129 1 OPERATOR_ROUTINE_TABLE (RPG_SUB_TABLE,
5015 5130 1 OPERATOR_ROUTINE (P, P, P, SUB_P_P));
5016 5131 1
5017 5132 1 ! Define the Operator Routine Table for RPG multiplication.
5018 5133 1
5019 P 5134 1 OPERATOR_ROUTINE_TABLE (RPG_MUL_TABLE,
5020 5135 1 OPERATOR_ROUTINE (P, P, P, MUL_P_P));
5021 5136 1
5022 5137 1 ! Define the Operator Routine Table for RPG division.
5023 5138 1
5024 P 5139 1 OPERATOR_ROUTINE_TABLE (RPG_DIV_TABLE,
5025 5140 1 OPERATOR_ROUTINE (P, P, P, DIV_P_P));
5026 5141 1
5027 5142 1 ! Define the Operator Routine Table for RPG unary plus.
5028 5143 1
5029 P 5144 1 OPERATOR_ROUTINE_TABLE (RPG_UNARY_PLUS_TABLE,
5030 P 5145 1
5031 P 5146 1 ! The following are not language dependent types. This is needed for DEBUG
5032 P 5147 1 types. for example, DEP/QUAD L= +1.
5033 P 5148 1
5034 P 5149 1 OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
5035 P 5150 1 OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
5036 P 5151 1 OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
5037 P 5152 1 OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
5038 P 5153 1 OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
5039 P 5154 1 OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
5040 P 5155 1 OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
5041 P 5156 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
5042 P 5157 1 OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
5043 P 5158 1
5044 5159 1 OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P));
5045 5160 1
5046 5161 1 ! Define the Operator Routine Table for RPG unary minus.
5047 5162 1
5048 P 5163 1 OPERATOR_ROUTINE_TABLE (RPG_UNARY_MINUS_TABLE,
5049 P 5164 1
```

```
5050 P 5165 1 ! The following are not language dependent types. This is needed for DEBUG
5051 P 5166 1 ! types. For example, DEP/QUAD L= +1.
5052 P 5167 1 !
5053 P 5168 1 ! OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
5054 P 5169 1 ! OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
5055 P 5170 1 ! OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
5056 P 5171 1 ! OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
5057 P 5172 1 ! OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
5058 P 5173 1 ! OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
5059 P 5174 1 ! OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
5060 P 5175 1 ! OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
5061 P 5176 1 ! OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
5062 P 5177 1 !
5063 P 5178 1 ! OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P));
5064 P 5179 1 !
5065 P 5180 1 ! Define the Operator Routine Table for RPG =.
5066 P 5181 1 !
5067 P 5182 1 ! OPERATOR_ROUTINE_TABLE (RPG_EQL_TABLE,
5068 P 5183 1 ! OPERATOR_ROUTINE (T, T, TF, EQL_T_T),
5069 P 5184 1 ! OPERATOR_ROUTINE (P, P, TF, EQL_P_P));
5070 P 5185 1 !
5071 P 5186 1 ! Define the Operator Routine Table for RPG NOT =.
5072 P 5187 1 !
5073 P 5188 1 ! OPERATOR_ROUTINE_TABLE (RPG_NEQ_TABLE,
5074 P 5189 1 ! OPERATOR_ROUTINE (T, T, TF, NEQ_T_T),
5075 P 5190 1 ! OPERATOR_ROUTINE (P, P, TF, NEQ_P_P));
5076 P 5191 1 !
5077 P 5192 1 ! Define the Operator Routine Table for RPG >.
5078 P 5193 1 !
5079 P 5194 1 ! OPERATOR_ROUTINE_TABLE (RPG_GTR_TABLE,
5080 P 5195 1 ! OPERATOR_ROUTINE (T, T, TF, GTR_T_T),
5081 P 5196 1 ! OPERATOR_ROUTINE (P, P, TF, GTR_P_P));
5082 P 5197 1 !
5083 P 5198 1 ! Define the Operator Routine Table for RPG NOT <.
5084 P 5199 1 !
5085 P 5200 1 ! OPERATOR_ROUTINE_TABLE (RPG_GEQ_TABLE,
5086 P 5201 1 ! OPERATOR_ROUTINE (T, T, TF, GEQ_T_T),
5087 P 5202 1 ! OPERATOR_ROUTINE (P, P, TF, GEQ_P_P));
5088 P 5203 1 !
5089 P 5204 1 ! Define the Operator Routine Table for RPG <.
5090 P 5205 1 !
5091 P 5206 1 ! OPERATOR_ROUTINE_TABLE (RPG_LSS_TABLE,
5092 P 5207 1 ! OPERATOR_ROUTINE (T, T, TF, LSS_T_T),
5093 P 5208 1 ! OPERATOR_ROUTINE (P, P, TF, LSS_P_P));
5094 P 5209 1 !
5095 P 5210 1 ! Define the Operator Routine Table for RPG NOT >.
5096 P 5211 1 !
5097 P 5212 1 ! OPERATOR_ROUTINE_TABLE (RPG_LEQ_TABLE,
5098 P 5213 1 ! OPERATOR_ROUTINE (T, T, TF, LEQ_T_T),
5099 P 5214 1 ! OPERATOR_ROUTINE (P, P, TF, LEQ_P_P));
5100 P 5215 1 !
5101 P 5216 1 ! Define the Operator Routine Table for RPG NOT.
5102 P 5217 1 !
5103 P 5218 1 ! OPERATOR_ROUTINE_TABLE (RPG_NOT_TABLE,
5104 P 5219 1 ! OPERATOR_ROUTINE (TF, TF, TF, NOT_L));
5105 P 5220 1 !
5106 P 5221 1 ! Define the Operator Routine Table for RPG AND.
```

```
5107 5222 1 !
5108 P 5223 1 OPERATOR ROUTINE TABLE (RPG AND TABLE,
5109 5224 1 OPERATOR_ROUTINE (TF, TF, TF, AND_L_L));
5110 5225 1
5111 5226 1 ! Define the Operator Routine Table for RPG OR.
5112 5227 1 !
5113 P 5228 1 OPERATOR ROUTINE TABLE (RPG OR TABLE,
5114 5229 1 OPERATOR_ROUTINE (TF, TF, TF, OR_L_L));
5115 5230 1
5116 5231 1
5117 5232 1 ! Define the Operator Information Table for RPG.
5118 5233 1 !
5119 P 5234 1 OPERATOR INFO TABLE (RPG_OPINFO_TABLE,
5120 P 5235 1 OPERATOR_INFO_ENTRY
5121 P 5236 1 (ADD, RPG_ADD_TABLE, RPG_HIER_TABLE, TABLEBASE),
5122 P 5237 1 OPERATOR_INFO_ENTRY
5123 P 5238 1 (SUBTRACT, RPG_SUB_TABLE, RPG_HIER_TABLE, TABLEBASE),
5124 P 5239 1 OPERATOR_INFO_ENTRY
5125 P 5240 1 (MULTIPLY, RPG_MUL_TABLE, RPG_HIER_TABLE, TABLEBASE),
5126 P 5241 1 OPERATOR_INFO_ENTRY
5127 P 5242 1 (DIVIDE, RPG_DIV_TABLE, RPG_HIER_TABLE, TABLEBASE),
5128 P 5243 1 OPERATOR_INFO_ENTRY
5129 P 5244 1 (UNARY PLUS, RPG_UNARY_PLUS_TABLE, RPG_HIER_TABLE, TABLEBASE),
5130 P 5245 1 OPERATOR_INFO_ENTRY
5131 P 5246 1 (UNARY MINUS, RPG_UNARY_MINUS_TABLE, RPG_HIER_TABLE, TABLEBASE),
5132 P 5247 1 OPERATOR_INFO_ENTRY
5133 P 5248 1 (EQUAL, RPG_EQL_TABLE, RPG_HIER_TABLE, TABLEBASE),
5134 P 5249 1 OPERATOR_INFO_ENTRY
5135 P 5250 1 (NOT_EQUAL, RPG_NEQ_TABLE, RPG_HIER_TABLE, TABLEBASE),
5136 P 5251 1 OPERATOR_INFO_ENTRY
5137 P 5252 1 (GTR_THAN, RPG_GTR_TABLE, RPG_HIER_TABLE, TABLEBASE),
5138 P 5253 1 OPERATOR_INFO_ENTRY
5139 P 5254 1 (GTR_EQUAL, RPG_GEQ_TABLE, RPG_HIER_TABLE, TABLEBASE),
5140 P 5255 1 OPERATOR_INFO_ENTRY
5141 P 5256 1 (LSS_THAN, RPG_LSS_TABLE, RPG_HIER_TABLE, TABLEBASE),
5142 P 5257 1 OPERATOR_INFO_ENTRY
5143 P 5258 1 (LSS_EQUAL, RPG_LEQ_TABLE, RPG_HIER_TABLE, TABLEBASE),
5144 P 5259 1 OPERATOR_INFO_ENTRY
5145 P 5260 1 (NOT, RPG_NOT_TABLE, RPG_HIER_TABLE, TABLEBASE),
5146 P 5261 1 OPERATOR_INFO_ENTRY
5147 P 5262 1 (AND, RPG_AND_TABLE, RPG_HIER_TABLE, TABLEBASE),
5148 P 5263 1 OPERATOR_INFO_ENTRY
5149 P 5264 1 (OR, RPG_OR_TABLE, RPG_HIER_TABLE, TABLEBASE),
5150 P 5265 1 OPERATOR_INFO_ENTRY
5151 P 5266 1 (CONVERT, TABLEBASE, RPG_HIERD_TABLE, TABLEBASE),
5152 P 5267 1 OPERATOR_INFO_ENTRY
5153 P 5268 1 (DEPOSIT, TABLEBASE, RPG_HIERD_TABLE, TABLEBASE),
5154 P 5269 1 OPERATOR_INFO_ENTRY
5155 5270 1 (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE));
5156 5271 1
```

```
5158 5272 1
5159 5273 1
5160 5274 1
5161 5275 1
5162 5276 1
5163 5277 1
5164 5278 1
5165 5279 1
5166 5280 1
5167 5281 1
5168 5282 1
5169 5283 1
5170 5284 1
5171 5285 1
5172 5286 1
5173 5287 1
5174 5288 1
5175 5289 1
5176 5290 1
5177 5291 1
5178 5292 1
5179 5293 1
5180 5294 1
5181 5295 1
5182 P 5296 1
5183 5297 1
5184 5298 1
5185 5299 1
5186 5300 1
5187 5301 1
5188 5302 1
5189 5303 1
5190 5304 1
5191 5305 1
5192 5306 1
5193 P 5307 1
5194 P 5308 1
5195 P 5309 1
5196 P 5310 1
5197 P 5311 1
5198 P 5312 1
5199 P 5313 1
5200 P 5314 1
5201 P 5315 1
5202 P 5316 1
5203 P 5317 1
5204 P 5318 1
5205 P 5319 1
5206 P 5320 1
5207 P 5321 1
5208 P 5322 1
5209 P 5323 1
5210 P 5324 1
5211 P 5325 1
5212 P 5326 1
5213 5327 1
5214 5328 1
```

UNKNOWN OPERATOR INFORMATION TABLES

This section contains the Operator Routine and Type tables needed to evaluate expressions in the UNKNOWN language.

The purpose of 'UNKNOWN' is to provide some level of support for other languages by allowing SET LANG UNKNOWN. E.g., if somebody is writing a compiler for a language other than one which DEBUG supports such as JOVIAL, HAL, SIMULA, ...., they can get some level of DEBUG support by generating DST and saying SET LANG UNK

So, the guiding principal below is to allow as many data types as possible and be very permissive about conversions between data types. This will increase the chance of covering the allowed operations and data types in a given language. (Although it probably means we will be allowing many things that the language does not allow).

Define the Type Conversion Information Table for UNKNOWN.  
There are no initial type mappings or exceptions to language-specific conversion rules.

CONVERSION\_INFO\_TABLE (UNKNOWN\_CVTINFO\_TABLE,  
CONVERSION\_INFO\_ENTRY (TABLEBASE, TABLEBASE));

Define the Type Hierarchy Table for UNKNOWN.  
This allows most integer-based types to be converted to L,  
and L can then be converted up to F, D, G, and H.  
The complex types are not included here - they increase the size  
of all the tables, and most languages do not support complex  
arithmetic, so it was not deemed worth it.

TYPE\_HIERARCHY\_TABLE (UNKNOWN\_HIER\_TABLE,  
TYPE\_GRAPH\_EDGE (TF, L),  
TYPE\_GRAPH\_EDGE (PTR, L),  
TYPE\_GRAPH\_EDGE (TPTR, L),  
TYPE\_GRAPH\_EDGE (ENUM, L),  
TYPE\_GRAPH\_EDGE (BU, WU),  
TYPE\_GRAPH\_EDGE (WU, LU),  
TYPE\_GRAPH\_EDGE (LU, L),  
TYPE\_GRAPH\_EDGE (B, W),  
TYPE\_GRAPH\_EDGE (W, L),  
TYPE\_GRAPH\_EDGE (L, F),  
TYPE\_GRAPH\_EDGE (F, D),  
TYPE\_GRAPH\_EDGE (F, G),  
TYPE\_GRAPH\_EDGE (F, FC),  
TYPE\_GRAPH\_EDGE (D, H),  
TYPE\_GRAPH\_EDGE (D, DC),  
TYPE\_GRAPH\_EDGE (G, H),  
TYPE\_GRAPH\_EDGE (G, GC),  
TYPE\_GRAPH\_EDGE (FC, DC),  
TYPE\_GRAPH\_EDGE (FC, GC),  
0);

```
5215 5329 1
5216 5330 1
5217 5331 1
5218 5332 1
5219 5333 1
5220 5334 1
5221 P 5335 1
5222 P 5336 1
5223 P 5337 1
5224 P 5338 1
5225 P 5339 1
5226 P 5340 1
5227 P 5341 1
5228 P 5342 1
5229 P 5343 1
5230 P 5344 1
5231 P 5345 1
5232 P 5346 1
5233 P 5347 1
5234 P 5348 1
5235 P 5349 1
5236 P 5350 1
5237 P 5351 1
5238 P 5352 1
5239 P 5353 1
5240 P 5354 1
5241 5355 1
5242 5356 1
5243 5357 1
5244 5358 1
5245 5359 1
5246 P 5360 1
5247 P 5361 1
5248 P 5362 1
5249 P 5363 1
5250 P 5364 1
5251 5365 1
5252 5366 1
5253 5367 1
5254 5368 1
5255 5369 1
5256 5370 1
5257 5371 1
5258 5372 1
5259 5373 1
5260 P 5374 1
5261 P 5375 1
5262 P 5376 1
5263 P 5377 1
5264 P 5378 1
5265 P 5379 1
5266 P 5380 1
5267 P 5381 1
5268 5382 1
5269 5383 1
5270 5384 1
5271 5385 1

! Define the Type Hierarchy Table for UNKNOWN deposit.
! This is a circular graph which allows any type to be deposited
! into any other (even though DBG$CVT_DX_DX may not allow all
! of the combinations.)
TYPE_HIERARCHY_TABLE (UNKNOWN_HIERD_TABLE,
    TYPE_GRAPH_EDGE (BU, B),
    TYPE_GRAPH_EDGE (B, WU),
    TYPE_GRAPH_EDGE (WU, W),
    TYPE_GRAPH_EDGE (W, LU),
    TYPE_GRAPH_EDGE (LU, L),
    TYPE_GRAPH_EDGE (L, F),
    TYPE_GRAPH_EDGE (F, D),
    TYPE_GRAPH_EDGE (F, G),
    TYPE_GRAPH_EDGE (D, H),
    TYPE_GRAPH_EDGE (G, H),
    TYPE_GRAPH_EDGE (H, FC),
    TYPE_GRAPH_EDGE (FC, DC),
    TYPE_GRAPH_EDGE (FC, GC),
    TYPE_GRAPH_EDGE (DC, ENUM),
    TYPE_GRAPH_EDGE (GC, ENUM),
    TYPE_GRAPH_EDGE (ENUM, TPTR),
    TYPE_GRAPH_EDGE (TPTR, PTR),
    TYPE_GRAPH_EDGE (PTR, TF),
    TYPE_GRAPH_EDGE (TF, BU),
    0);

! Define the Type Incompatibility Table for language UNKNOWN.
! D and G arithmetic is always incompatible.
TYPE_INCOMP_TABLE (UNKNOWN_INCOMP_TABLE,
    TYPE_GRAPH_EDGE (D, G),
    TYPE_GRAPH_EDGE (D, GC),
    TYPE_GRAPH_EDGE (G, DC),
    TYPE_GRAPH_EDGE (DC, GC),
    0);

! Allow arithmetic on any of the numeric types.
! Other types (PTR, TPTR, ENUM, unsigned types) can be converted to L before doing
! any arithmetic.

! Define the Operator Routine Table for UNKNOWN addition.
OPERATOR_ROUTINE_TABLE (UNKNOWN_ADD_TABLE,
    OPERATOR_ROUTINE (L, L, L, ADD_L_L),
    OPERATOR_ROUTINE (F, F, F, ADD_F_F),
    OPERATOR_ROUTINE (D, D, D, ADD_D_D),
    OPERATOR_ROUTINE (G, G, G, ADD_G_G),
    OPERATOR_ROUTINE (H, H, H, ADD_H_H),
    OPERATOR_ROUTINE (FC, FC, FC, ADD_FC_FC),
    OPERATOR_ROUTINE (DC, DC, DC, ADD_DC_DC),
    OPERATOR_ROUTINE (GC, GC, GC, ADD_GC_GC));

! Define the Operator Routine Table for UNKNOWN subtraction.
```

```
5272 5386 1 !
5273 P 5387 1 OPERATOR_ROUTINE_TABLE (UNKNOWN SUB TABLE,
5274 P 5388 1 OPERATOR_ROUTINE (L, L, L, SUB_C_L),
5275 P 5389 1 OPERATOR_ROUTINE (F, F, F, SUB_F_F),
5276 P 5390 1 OPERATOR_ROUTINE (D, D, D, SUB_D_D),
5277 P 5391 1 OPERATOR_ROUTINE (G, G, G, SUB_G_G),
5278 P 5392 1 OPERATOR_ROUTINE (H, H, H, SUB_H_H),
5279 P 5393 1 OPERATOR_ROUTINE (FC, FC, FC, SUB_FC_FC),
5280 P 5394 1 OPERATOR_ROUTINE (DC, DC, DC, SUB_DC_DC),
5281 5395 1 OPERATOR_ROUTINE (GC, GC, GC, SUB_GC_GC));
5282 5396 1
5283 5397 1
5284 5398 1 ! Define the Operator Routine Table for UNKNOWN multiplication.
5285 5399 1
5286 P 5400 1 OPERATOR_ROUTINE_TABLE (UNKNOWN MUL TABLE,
5287 P 5401 1 OPERATOR_ROUTINE (L, L, L, MUL_C_L),
5288 P 5402 1 OPERATOR_ROUTINE (F, F, F, MUL_F_F),
5289 P 5403 1 OPERATOR_ROUTINE (D, D, D, MUL_D_D),
5290 P 5404 1 OPERATOR_ROUTINE (G, G, G, MUL_G_G),
5291 P 5405 1 OPERATOR_ROUTINE (H, H, H, MUL_H_H),
5292 P 5406 1 OPERATOR_ROUTINE (FC, FC, FC, MUL_FC_FC),
5293 P 5407 1 OPERATOR_ROUTINE (DC, DC, DC, MUL_DC_DC),
5294 5408 1 OPERATOR_ROUTINE (GC, GC, GC, MUL_GC_GC));
5295 5409 1
5296 5410 1
5297 5411 1 ! Define the Operator Routine Table for UNKNOWN division.
5298 5412 1
5299 P 5413 1 OPERATOR_ROUTINE_TABLE (UNKNOWN DIV TABLE,
5300 P 5414 1 OPERATOR_ROUTINE (L, L, L, DIV_C_L),
5301 P 5415 1 OPERATOR_ROUTINE (F, F, F, DIV_F_F),
5302 P 5416 1 OPERATOR_ROUTINE (D, D, D, DIV_D_D),
5303 P 5417 1 OPERATOR_ROUTINE (G, G, G, DIV_G_G),
5304 P 5418 1 OPERATOR_ROUTINE (H, H, H, DIV_H_H),
5305 P 5419 1 OPERATOR_ROUTINE (FC, FC, FC, DIV_FC_FC),
5306 P 5420 1 OPERATOR_ROUTINE (DC, DC, DC, DIV_DC_DC),
5307 5421 1 OPERATOR_ROUTINE (GC, GC, GC, DIV_GC_GC));
5308 5422 1
5309 5423 1
5310 5424 1 ! Define the Operator Routine Table for UNKNOWN unary plus.
5311 5425 1
5312 P 5426 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_UNARY_PLUS_TABLE,
5313 P 5427 1
5314 P 5428 1 ! The following are not language dependent types. This is needed for DEBUG
5315 P 5429 1 ! types. For example, DEP/QUAD L= +1.
5316 P 5430 1
5317 P 5431 1 OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
5318 P 5432 1 OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
5319 P 5433 1 OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
5320 P 5434 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
5321 P 5435 1 OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
5322 P 5436 1
5323 P 5437 1 OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
5324 P 5438 1 OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
5325 P 5439 1 OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
5326 P 5440 1 OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
5327 P 5441 1 OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
5328 P 5442 1 OPERATOR_ROUTINE (FC, FC, FC, UNARY_PLUS_FC).
```

```
5329 P 5443 1 OPERATOR_ROUTINE (DC, DC, DC, UNARY_PLUS_DC),
5330 5444 1 OPERATOR_ROUTINE (GC, GC, GC, UNARY_PLUS_GC));
5331 5445 1
5332 5446 1
5333 5447 1 ! Define the Operator Routine Table for UNKNOWN unary minus.
5334 5448 1
5335 P 5449 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_UNARY_MINUS_TABLE,
5336 P 5450 1
5337 P 5451 1 ! The following are not language dependent types. This is needed for DEBUG
5338 P 5452 1 ! types. For example, DEP/QUAD L= +1.
5339 P 5453 1
5340 P 5454 1 OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
5341 P 5455 1 OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
5342 P 5456 1 OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
5343 P 5457 1 OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
5344 P 5458 1 OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
5345 P 5459 1
5346 P 5460 1 OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
5347 P 5461 1 OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
5348 P 5462 1 OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
5349 P 5463 1 OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
5350 P 5464 1 OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
5351 P 5465 1 OPERATOR_ROUTINE (FC, FC, FC, UNARY_MINUS_FC),
5352 P 5466 1 OPERATOR_ROUTINE (DC, DC, DC, UNARY_MINUS_DC),
5353 5467 1 OPERATOR_ROUTINE (GC, GC, GC, UNARY_MINUS_GC));
5354 5468 1
5355 5469 1
5356 5470 1 ! Define the Operator Routine Table for UNKNOWN exponentiation.
5357 5471 1 ! We special-case the mixed cases F**L, D**L, G**L, and H**L because
5358 5472 1 ! we do not want the exponent converted to float there. Note - this
5359 5473 1 ! is the same table as for FORTRAN.
5360 5474 1
5361 P 5475 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_POWER_TABLE,
5362 P 5476 1 OPERATOR_ROUTINE (W, W, W, POWER_Q_W),
5363 P 5477 1 OPERATOR_ROUTINE (L, L, L, POWER_L_L),
5364 P 5478 1 OPERATOR_ROUTINE (F, L, F, POWER_F_L),
5365 P 5479 1 OPERATOR_ROUTINE (D, L, D, POWER_D_L),
5366 P 5480 1 OPERATOR_ROUTINE (G, L, G, POWER_G_L),
5367 P 5481 1 OPERATOR_ROUTINE (H, L, H, POWER_H_L),
5368 P 5482 1 OPERATOR_ROUTINE (FC, L, FC, POWER_FC_L),
5369 P 5483 1 OPERATOR_ROUTINE (DC, L, DC, POWER_DC_L),
5370 P 5484 1 OPERATOR_ROUTINE (GC, L, GC, POWER_GC_L),
5371 P 5485 1 OPERATOR_ROUTINE (F, F, F, POWER_F_F),
5372 P 5486 1 OPERATOR_ROUTINE (D, F, D, POWER_D_F),
5373 P 5487 1 OPERATOR_ROUTINE (F, D, D, POWER_F_D),
5374 P 5488 1 OPERATOR_ROUTINE (D, D, D, POWER_D_D),
5375 P 5489 1 OPERATOR_ROUTINE (G, G, G, POWER_G_G),
5376 P 5490 1 OPERATOR_ROUTINE (H, H, H, POWER_H_H),
5377 P 5491 1 OPERATOR_ROUTINE (FC, FC, FC, POWER_FC_FC),
5378 P 5492 1 OPERATOR_ROUTINE (DC, DC, DC, POWER_DC_DC),
5379 5493 1 OPERATOR_ROUTINE (GC, GC, GC, POWER_GC_GC));
5380 5494 1
5381 5495 1
5382 5496 1 ! The relationals are defined on strings and on numeric types.
5383 5497 1 ! We do not specially handle types like ENUM, TPTR, and so on, as
5384 5498 1 ! in PASCAL - here we just convert them to integer and do integer
5385 5499 1 ! comparison. The intent is to be looser about type-checking
```

```
5386      5500 1 ! than is PASCAL.
5387      5501 1
5388      5502 1 ! Define the Operator Routine Table for UNKNOWN equal.
5389      5503 1
5390      P 5504 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_EQL_TABLE,
5391      P 5505 1   OPERATOR_ROUTINE (T, T, TF, EQL_T_T),
5392      P 5506 1   OPERATOR_ROUTINE (L, L, TF, EQL_L_L),
5393      P 5507 1   OPERATOR_ROUTINE (F, F, TF, EQL_F_F),
5394      P 5508 1   OPERATOR_ROUTINE (D, D, TF, EQL_D_D),
5395      P 5509 1   OPERATOR_ROUTINE (G, G, TF, EQL_G_G),
5396      P 5510 1   OPERATOR_ROUTINE (H, H, TF, EQL_H_H),
5397      P 5511 1   OPERATOR_ROUTINE (FC, FC, TF, EQL_FC_FC),
5398      P 5512 1   OPERATOR_ROUTINE (DC, DC, TF, EQL_DC_DC),
5399      5513 1   OPERATOR_ROUTINE (GC, GC, TF, EQL_GC_GC));
5400      5514 1
5401      5515 1 ! Define the Operator Routine Table for UNKNOWN not-equal.
5402      5516 1
5403      P 5517 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_NEQ_TABLE,
5404      P 5518 1   OPERATOR_ROUTINE (T, T, TF, NEQ_T_T),
5405      P 5519 1   OPERATOR_ROUTINE (L, L, TF, NEQ_L_L),
5406      P 5520 1   OPERATOR_ROUTINE (F, F, TF, NEQ_F_F),
5407      P 5521 1   OPERATOR_ROUTINE (D, D, TF, NEQ_D_D),
5408      P 5522 1   OPERATOR_ROUTINE (G, G, TF, NEQ_G_G),
5409      P 5523 1   OPERATOR_ROUTINE (H, H, TF, NEQ_H_H),
5410      P 5524 1   OPERATOR_ROUTINE (FC, FC, TF, NEQ_FC_FC),
5411      P 5525 1   OPERATOR_ROUTINE (DC, DC, TF, NEQ_DC_DC),
5412      5526 1   OPERATOR_ROUTINE (GC, GC, TF, NEQ_GC_GC));
5413      5527 1
5414      5528 1
5415      5529 1 ! Define the Operator Routine Table for UNKNOWN less than.
5416      5530 1
5417      P 5531 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_LSS_THAN_TABLE,
5418      P 5532 1   OPERATOR_ROUTINE (T, T, TF, LSS_T_T),
5419      P 5533 1   OPERATOR_ROUTINE (L, L, TF, LSS_L_L),
5420      P 5534 1   OPERATOR_ROUTINE (F, F, TF, LSS_F_F),
5421      P 5535 1   OPERATOR_ROUTINE (D, D, TF, LSS_D_D),
5422      P 5536 1   OPERATOR_ROUTINE (G, G, TF, LSS_G_G),
5423      5537 1   OPERATOR_ROUTINE (H, H, TF, LSS_H_H));
5424      5538 1
5425      5539 1 ! Define the Operator Routine Table for UNKNOWN greater than.
5426      5540 1
5427      P 5541 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_GTR_THAN_TABLE,
5428      P 5542 1   OPERATOR_ROUTINE (T, T, TF, GTR_T_T),
5429      P 5543 1   OPERATOR_ROUTINE (L, L, TF, GTR_L_L),
5430      P 5544 1   OPERATOR_ROUTINE (F, F, TF, GTR_F_F),
5431      P 5545 1   OPERATOR_ROUTINE (D, D, TF, GTR_D_D),
5432      P 5546 1   OPERATOR_ROUTINE (G, G, TF, GTR_G_G),
5433      5547 1   OPERATOR_ROUTINE (H, H, TF, GTR_H_H));
5434      5548 1
5435      5549 1
5436      5550 1 ! Define the Operator Routine Table for UNKNOWN less than or equal.
5437      5551 1
5438      P 5552 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_LEQ_TABLE,
5439      P 5553 1   OPERATOR_ROUTINE (T, T, TF, LEQ_T_T),
5440      P 5554 1   OPERATOR_ROUTINE (L, L, TF, LEQ_L_L),
5441      P 5555 1   OPERATOR_ROUTINE (F, F, TF, LEQ_F_F),
5442      P 5556 1   OPERATOR_ROUTINE (D, D, TF, LEQ_D_D).
```

```
5443 P 5557 1 OPERATOR_ROUTINE (G, G, TF, LEQ_G_G);
5444 5558 1 OPERATOR_ROUTINE (H, H, TF, LEQ_H_H);
5445 5559 1
5446 5560 1 ! Define the Operator Routine Table for UNKNOWN greater than or equal.
5447 5561 1
5448 P 5562 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_GEQ_TABLE,
5449 P 5563 1 OPERATOR_ROUTINE (T, T, TF, GEQ_T_T),
5450 P 5564 1 OPERATOR_ROUTINE (L, L, TF, GEQ_L_L),
5451 P 5565 1 OPERATOR_ROUTINE (F, F, TF, GEQ_F_F),
5452 P 5566 1 OPERATOR_ROUTINE (D, D, TF, GEQ_D_D),
5453 P 5567 1 OPERATOR_ROUTINE (G, G, TF, GEQ_G_G),
5454 5568 1 OPERATOR_ROUTINE (H, H, TF, GEQ_H_H);
5455 5569 1
5456 5570 1 ! The logical operators AND, OR, NOT, EQV and XOR are defined only
5457 5571 1 ! on integers. Other types such as Tf can be converted to Boolean.
5458 5572 1
5459 5573 1 ! Define the Operator Routine Table for UNKNOWN AND.
5460 5574 1
5461 P 5575 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_AND_TABLE,
5462 5576 1 OPERATOR_ROUTINE (L, L, L, BIT_AND_L_L));
5463 5577 1
5464 5578 1 ! Define the Operator Routine Table for UNKNOWN OR.
5465 5579 1
5466 P 5580 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_OR_TABLE,
5467 5581 1 OPERATOR_ROUTINE (L, L, L, BIT_OR_L_L));
5468 5582 1
5469 5583 1 ! Define the Operator Routine Table for UNKNOWN XOR.
5470 5584 1
5471 P 5585 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_XOR_TABLE,
5472 5586 1 OPERATOR_ROUTINE (L, L, L, BIT_XOR_L_L));
5473 5587 1
5474 5588 1 ! Define the Operator Routine Table for UNKNOWN EQV.
5475 5589 1
5476 P 5590 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_EQV_TABLE,
5477 5591 1 OPERATOR_ROUTINE (L, L, L, BIT_EQV_L_L));
5478 5592 1
5479 5593 1 ! Define the Operator Routine Table for UNKNOWN NOT.
5480 5594 1
5481 P 5595 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_NOT_TABLE,
5482 5596 1 OPERATOR_ROUTINE (L, L, L, BIT_NOT_L));
5483 5597 1
5484 5598 1 ! Define the Operator Routine Table for UNKNOWN concatenate.
5485 5599 1
5486 P 5600 1 OPERATOR_ROUTINE_TABLE (UNKNOWN_CONCATENATE_TABLE,
5487 5601 1 OPERATOR_ROUTINE (T, T, T, CONCAT_T_T));
5488 5602 1
5489 5603 1 ! Define the Operator Information Table for UNKNOWN.
5490 5604 1
5491 P 5605 1 OPERATOR_INFO_TABLE (UNKNOWN_OPINFO_TABLE,
5492 P 5606 1
5493 P 5607 1 ! All of the arithmetic, logical, and relational operators use
5494 P 5608 1 ! the same hierarchy table and the same incompatibility table.
5495 P 5609 1
5496 P 5610 1 ! Arithmetic.
5497 P 5611 1
5498 P 5612 1 OPERATOR_INFO_ENTRY
5499 P 5613 1 (ADD, UNKNOWN_ADD_TABLE,
```

```
5500 P 5614 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5501 P 5615 1 OPERATOR_INFO_ENTRY
5502 P 5616 1 (SUBTRACT, UNKNOWN_SUB_TABLE,
5503 P 5617 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5504 P 5618 1 OPERATOR_INFO_ENTRY
5505 P 5619 1 (MULTIPLY, UNKNOWN_MUL_TABLE,
5506 P 5620 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5507 P 5621 1 OPERATOR_INFO_ENTRY
5508 P 5622 1 (DIVIDE, UNKNOWN_DIV_TABLE,
5509 P 5623 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5510 P 5624 1 OPERATOR_INFO_ENTRY
5511 P 5625 1 (UNARY_PLUS, UNKNOWN_UNARY_PLUS_TABLE,
5512 P 5626 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5513 P 5627 1 OPERATOR_INFO_ENTRY
5514 P 5628 1 (UNARY_MINUS, UNKNOWN_UNARY_MINUS_TABLE,
5515 P 5629 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5516 P 5630 1 OPERATOR_INFO_ENTRY
5517 P 5631 1 (POWER_OF, UNKNOWN_POWER_TABLE,
5518 P 5632 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5519 P 5633 1
5520 P 5634 1 ! Relational
5521 P 5635 1 !
5522 P 5636 1 OPERATOR_INFO_ENTRY
5523 P 5637 1 (EQUAL, UNKNOWN_EQL_TABLE,
5524 P 5638 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5525 P 5639 1 OPERATOR_INFO_ENTRY
5526 P 5640 1 (NOT_EQUAL, UNKNOWN_NEQ_TABLE,
5527 P 5641 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5528 P 5642 1 OPERATOR_INFO_ENTRY
5529 P 5643 1 (LSS_THAN, UNKNOWN_LSS_THAN_TABLE,
5530 P 5644 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5531 P 5645 1 OPERATOR_INFO_ENTRY
5532 P 5646 1 (GTR_THAN, UNKNOWN_GTR_THAN_TABLE,
5533 P 5647 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5534 P 5648 1 OPERATOR_INFO_ENTRY
5535 P 5649 1 (LSS_EQUAL, UNKNOWN_LEQ_TABLE,
5536 P 5650 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5537 P 5651 1 OPERATOR_INFO_ENTRY
5538 P 5652 1 (GTR_EQUAL, UNKNOWN_GEQ_TABLE,
5539 P 5653 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5540 P 5654 1
5541 P 5655 1 ! Logical
5542 P 5656 1 !
5543 P 5657 1 OPERATOR_INFO_ENTRY
5544 P 5658 1 (AND, UNKNOWN_AND_TABLE,
5545 P 5659 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5546 P 5660 1 OPERATOR_INFO_ENTRY
5547 P 5661 1 (OR, UNKNOWN_OR_TABLE,
5548 P 5662 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5549 P 5663 1 OPERATOR_INFO_ENTRY
5550 P 5664 1 (XOR, UNKNOWN_XOR_TABLE,
5551 P 5665 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5552 P 5666 1 OPERATOR_INFO_ENTRY
5553 P 5667 1 (EQV, UNKNOWN_EQV_TABLE,
5554 P 5668 1 UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
5555 P 5669 1 OPERATOR_INFO_ENTRY
5556 P 5670 1 (NOT, UNKNOWN_NOT_TABLE,
```

```

: 5557 P 5671 1      UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
: 5558 P 5672 1
: 5559 P 5673 1      ! String operations.
: 5560 P 5674 1
: 5561 P 5675 1      OPERATOR_INFO_ENTRY
: 5562 P 5676 1      (CONCATENATE, UNKNOWN_CONCATENATE_TABLE,
: 5563 P 5677 1      UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
: 5564 P 5678 1
: 5565 P 5679 1      ! CONVERT is used for things like converting subscripts.
: 5566 P 5680 1
: 5567 P 5681 1      OPERATOR_INFO_ENTRY
: 5568 P 5682 1      (CONVERT, TABLEBASE, UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
: 5569 P 5683 1
: 5570 P 5684 1      ! DEPOSIT is used in the DEPOSIT command.
: 5571 P 5685 1
: 5572 P 5686 1      OPERATOR_INFO_ENTRY
: 5573 P 5687 1      (DEPOSIT, TABLEBASE, UNKNOWN_HIERD_TABLE, UNKNOWN_INCOMP_TABLE),
: 5574 P 5688 1
: 5575 P 5689 1      ! IDENTITY is used at the end of an EVALUATE to turn the primary
: 5576 P 5690 1      ! into a value descriptor.
: 5577 P 5691 1
: 5578 P 5692 1      OPERATOR_INFO_ENTRY
: 5579   5693 1      (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE));
: 5580   5694 1
```

DBGEVALOP  
V04-000

6 3  
16-Sep-1984 00:32:25 VAX-11 Bliss-32 V4.0-742  
5-Sep-1984 21:54:24 [DEBUG.SRC]DBGEVALOP.B32;1

Page 107  
(21)

```
.. 5582      5695 1  | The following data structure holds a type pair in a word and is used
.. 5583      5696 1  | to search for the type pair in the Operator Routine Table.
.. 5584      5697 1  |
.. 5585      5698 1  | FIELD
.. 5586      5699 1  |     TYPE_PAIR_FLDS =
.. 5587      5700 1  |     SET
.. 5588      5701 1  |         BS_RIGHT_TYPE = [0,B0_];
.. 5589      5702 1  |         BS_LEFT_TYPE  = [0,B1_];
.. 5590      5703 1  |         WS_TYPE_PAIR  = [0,W0_];
.. 5591      5704 1  |     TES;
.. 5592      5705 1  |
.. 5593      5706 1  | MACRO
.. 5594      5707 1  |     TYPESPAIR = BLOCK [1, WORD] FIELD (TYPE_PAIR_FLDS) %;
.. 5595      5708 1  |
```

```
.. 5597      5709 1 ROUTINE AAA_DUMMY =  
.. 5598      5710 1  
.. 5599      5711 1 FUNCTION  
.. 5600      5712 1 This routine just causes the above tables to be dumped in the  
.. 5601      5713 1 listing.  
.. 5602      5714 1  
.. 5603      5715 2 BEGIN  
.. 5604      5716 2 0  
.. 5605      5717 1 END;
```

.TITLE DBGEVALOP  
.IDENT \V04-000\

.PSECT DBG\$PLIT,NOWRT, SHR, PIC,0

												45	53	41	42	00000	P.AAA:	.ASCII	\BASE\
												00000000	00000000	00004	ADA_CVTINFO	TABLE:			
													00	0000C		.LONG	0, 0		
														0000D		.BYTE	0		
													00000007	00010		.BLKB	3		
1C2B	1B0A	0B0A	2B0B	0A0B	0B04	0B03	0B07	0B02	0B06	00014	P.AAB:	.WORD	2054, 2050, 2055, 2051, 2052, 2568, -						
							0000	1C1B	1C0B	00028			11016, 2826, 6922, 7211, 7179, 7195, 0						
													0000000A	00030		.BLKB	2		
1C1B	1B0B	0B0A	0A2B	2B04	040B	0B03	0307	0702	0206	00034	P.AAC:	.WORD	518, 1794, 775, 2051, 1032, 11012, 2603, -						
0000	341C	341B	340B	340A	342B	340B	340E	342F	061C	00048			2826, 6923, 7195, 1564, 13359, 13326, -						
													00000016	0005C		.LONG	22		
07	00	00	00	BE	07	07	00	06	00	00	00	BD	06	06	00060	P.AAD:	.BYTE	6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -	
00	00	01	0A	09	09	00	15	00	00	00	00	F6	15	15	0006F			0, 0, 7, 0, 21, 21, -10, 0, 0, 0, 21, 0, -	
00	00	BF	08	08	00	1A	00	00	01	0C	1A	1A	00	09	0007E			9, 9, 10, 1, 0, 0, 9, 0, 26, 26, 12, 1, -	
00	C0	0A	0A	00	2B	00	00	01	10	2B	2B	00	08	00	0008D			0, 0, 26, 0, 8, 8, -65, 0, 0, 0, 8, 0, -	
C2	1B	1B	00	0B	00	00	00	C1	0B	0B	00	0A	00	00	0009C			43, 43, 16, 1, 0, 0, 43, 0, 10, 10, -64, -	
	00	1C	00	00	00	00	C3	1C	1C	00	1B	00	00	00	000AB			0, 0, 0, 10, 0, 11, 11, -63, 0, 0, 0, 11, -	
													00000016	00088		.LONG	22		
07	00	00	00	B2	07	07	00	06	00	00	00	B1	06	06	0008C	P.AAE:	.BYTE	6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -	
00	00	01	0B	09	09	00	15	00	00	00	00	F7	15	15	000CB			0, 0, 7, 0, 21, 21, -9, 0, 0, 0, 21, 0, -	
00	00	B3	0B	0B	00	1A	00	00	01	0D	1A	1A	00	09	000DA			9, 9, 11, 1, 0, 0, 9, 0, 26, 26, 13, 1, -	
00	B5	0A	0A	00	2B	00	00	01	11	2B	2B	00	08	00	000E9			0, 0, 26, 0, 8, 8, -77, 0, 0, 0, 8, 0, -	
B7	1B	1B	00	0B	00	00	00	B6	0B	0B	00	0A	00	00	000F8			43, 43, 17, 1, 0, 0, 43, 0, 10, 10, -75, -	
	00	1C	00	00	00	00	B8	1C	1C	00	1B	00	00	00	00107			0, 0, 0, 10, 0, 11, 11, -74, 0, 0, 0, 11, -	
													0000000C	00114		.LONG	12		
2B	00	00	01	12	2B	2B	00	0B	00	00	00	CA	0B	0B	00118	P.AAF:	.BYTE	8, 8, -54, 0, 0, 0, 8, 0, 43, 43, 18, 1, -	
00	00	00	CC	0B	0B	00	0A	00	00	00	00	0A	0A	00	00127			0, 0, 43, 0, 10, 10, -53, 0, 0, 0, 10, 0, -	
00	00	CE	1C	1C	00	1B	00	00	00	00	00	1B	00	0B	00136			11, 11, -52, 0, 0, 0, 11, 0, 27, 27, -51, -	
													0000000C	00148		.LONG	12		
2B	00	00	01	13	2B	2B	00	0B	00	00	00	05	0B	0B	0014C	P.AAG:	.BYTE	8, 8, 5, 0, 0, 0, 8, 0, 43, 43, 19, 1, 0, -	
00	00	00	0B	0B	0B	00	0A	00	00	00	07	0A	0A	00	0015B			0, 43, 0, 10, 10, 7, 0, 0, 0, 10, 0, 11, -	

Page 109  
(22)

00	00	0A	1C	1C	00	1B	00	00	00	09	1B	1B	00	0B	0016A	11	8	0	0	0	11	0	27	27	9	0	0	-
												00	1C	00	00179	0	27	0	28	28	10	0	0	0	28	0	0	-
2B	00	00	01	14	2B	2B	00	08	00	00	00	13	0B	0B	0017C	12	8	19	0	0	0	8	0	43	43	20	1	-
00	00	00	16	0B	0B	00	0A	00	00	00	15	0A	0A	00	00180	0	0	43	0	10	10	21	0	0	0	10	1	-
00	00	18	1C	1C	00	1B	00	00	00	17	1B	1B	00	0B	0018F	11	11	22	0	0	0	11	0	27	27	23	-	
												00	1C	00	0019E	0	0	0	27	0	28	28	24	0	0	0	28	0
												0000000C	00	0C	001AD	12	8	47	0	0	0	8	0	43	43	21	1	-
2B	00	00	01	15	2B	2B	00	08	00	00	00	2F	0B	0B	001B0	0	0	43	0	10	10	49	0	0	0	10	1	-
00	00	00	32	0B	0B	00	0A	00	00	00	31	0A	0A	00	001B4	11	11	50	0	0	0	11	0	27	27	51	-	
00	00	34	1C	1C	00	1B	00	00	00	33	1B	1B	00	0B	001C3	0	0	0	27	0	28	28	52	0	0	0	28	0
												00	1C	00	001D2	12	8	33	0	0	0	8	0	43	43	22	1	-
												0000000C	00	0C	001E1	0	0	43	0	10	10	35	0	0	0	10	1	-
2B	00	00	01	16	2B	2B	00	08	00	00	00	21	0B	0B	001E4	11	11	36	0	0	0	11	0	27	27	37	-	
00	00	00	24	0B	0B	00	0A	00	00	00	23	0A	0A	00	001E8	0	0	0	27	0	28	28	38	0	0	0	28	0
00	00	26	1C	1C	00	1B	00	00	00	25	1B	1B	00	0B	001F7	2	8	57	0	0	0	8	0	0	0	0	0	0
												00	1C	00	00206	2	8	59	0	0	0	8	0	0	0	0	0	0
												00000002	00	02	00215	10	8	65	0	0	0	8	0	10	10	73	0	-
												00000002	00	02	00218	0	0	10	0	11	11	76	0	0	0	11	0	-
												00000002	00	02	0021C	27	27	77	0	0	0	27	0	28	28	78	-	
												00000002	00	02	00224	0	0	0	28	0	0	0	0	0	0	0	0	-
												00000002	00	02	00228	40	40	-86	0	0	0	40	0	0	0	40	0	-
												0000000A	00	0A	00230	2	40	-90	0	0	0	40	0	0	0	40	0	-
0A	00	00	00	4																								

Page 110  
(22)

```
00000000 00000014 000003A0 0056C .BYTE 1
00000000 00000014 00000260 0056D .BYTE 0[19]
00000000 00000014 00000260 00580 .LONG 928, 20, 0
00000000 00000014 00000260 0058C .BYTE 1
00000000 00000014 00000260 0058D .BYTE 0[19]
00000000 00000014 00000260 005A0 .LONG 608, 20, 0
00000000 00000014 00000260 005AC .BYTE 1
00000000 00000014 00000260 005AD .BYTE 0[3]
00000000 00000014 00000260 005B0 .LONG 620, 20, 0
00000000 00000014 00000260 005BC .BYTE 1
00000000 00000014 00000260 005BD .BYTE 0[3]
00000000 00000014 00000278 005C0 .LONG 632, 20, 0
00000000 00000014 00000278 005CC .BYTE 1
00000000 00000014 00000278 005CD .BYTE 0[3]
00000000 00000014 00000284 005D0 .LONG 644, 20, 0
00000000 00000014 00000284 005DD .BYTE 1
00000000 00000014 00000428 005DD .BYTE 0[131]
00000000 00000014 00000428 00660 .LONG 1064, 20, 0
00000000 00000014 00000428 0066C .BYTE 1
00000000 00000014 00000428 0066D .BYTE 0[3]
00000000 00000014 0000021C 00670 .LONG 540, 20, 0
00000000 00000014 0000021C 0067C .BYTE 1
00000000 00000014 0000021C 0067D .BYTE 0[3]
00000000 00000014 00000228 00680 .LONG 552, 20, 0
00000000 00000014 00000228 0068C .BYTE 1
00000000 00000014 00000228 0068D .BYTE 0[83]
00000000 00000014 00000118 006E0 .LONG 280, 20, 0
00000000 00000014 00000118 006EC .BYTE 1
00000000 00000014 00000118 006ED .BYTE 0[99]
00000000 00000034 00000000 00750 .LONG 0, 52, 0
00000000 00000034 00000000 0075C .BYTE 1
00000000 00000034 00000000 0075D .BYTE 0[3]
00000000 00000034 00000000 00760 .LONG 0, 52, 0
00000000 00000034 00000000 0076C .BYTE 1
00000000 00000034 00000000 0076D .BYTE 0[67]
00000000 00000000 00000000 007B0 .LONG 0, 0, 0
00000000 00000000 00000000 007BC .BYTE 1
00000000 00000000 00000000 007BD .BLKB 179
00000000 00000000 00870 BASIC_CVT_INFO_TABLE:
00000000 00000000 00878 .LONG 0, 0
00000000 00000000 00879 .BYTE 0
00000000 00000000 0087C .BLKB 3
0000 1C1B 1C0B 1B0A 0B0A 1508 0A08 0807 0706 00880 P.AAY: .WORD 1798, 2055, 2568, 5384, 2826, 6922, 7179, -
00000000 00000000 00892 .BLKB 2
0000 0807 0706 00894 .LONG 1795, 0
0000 0807 0706 00898 P.AAZ: .WORD 1798, 2055, 0
00000000 00000000 0089E .BLKB 2
0000 061C 1C1B 1B0B 0B0A 0A15 1508 0807 0706 008A0 .LONG 1798, 2055, 5384, 2581, 2826, 6923, 7195, -
00000000 00000000 008A4 P.ABA: .WORD 1564, 0
00000000 00000000 008B6 .BLKB 2
00000000 00000000 008B8 .LONG 18
00000000 00000000 008BC P.ABB: .BYTE 6, 6, 1, 0, 0, 0, 6, 0, 7, 7, 3, 0, 0, 0, -
00000000 00000000 008CB .BYTE 7, 0, 8, 8, 5, 0, 0, 0, 8, 0, 10, 10, 7, -
00000000 00000000 008DA .BYTE 0, 0, 0, 10, 0, 11, 11, 8, 0, 0, 0, 11, -
```

DBGEVALOP  
V04-000

L 3  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 BLISS-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32;1

Page 112  
(22)

DBGEVALOP V04-000	00	F2	15	15	00	1C	00	00	00	0A	1C	1C	00	18	00	008E9
				00	0E	00	00	00	52	0E	0E	00	15	00	00	008F8
																00904
07	00	00	00	11	07	07	00	06	00	00	00	0F	06	06	00908	P.ABC: .LONG
00	00	00	15	0A	0A	00	08	00	00	00	13	08	08	00	00917	.BYTE
00	00	17	18	1B	00	08	00	00	00	16	0B	0B	00	0A	00926	
00	F3	15	15	00	1C	00	00	00	00	18	1C	1C	00	1B	00	00935
													00	15	00	00944
																00948
	00	00	00	2D	07	07	00	06	00	00	00	2B	06	06	0094C	P.ABD: .LONG
	00	00	31	0A	0A	00	08	00	00	00	2F	08	08	00	0095B	.BYTE
	00	33	1B	1B	00	08	00	00	00	32	0B	0B	00	0A	0096A	
F4	15	15	00	1C	00	00	00	00	34	1C	1C	00	1B	00	00979	
												00	15	00	00	00988
																0098C
	00	00	00	1F	07	07	00	06	00	00	00	1D	06	06	00990	P.ABE: .LONG
00	00	00	23	0A	0A	00	08	00	00	00	21	08	08	00	0099F	.BYTE
00	00	25	1B	1B	00	08	00	00	00	24	0B	0B	00	0A	009AE	
00	F5	15	15	00	1C	00	00	00	00	26	1C	1C	00	1B	00	009BD
												00	15	00	00	009CC
																009D0
1A	00	00	01	0C	1A	1A	00	09	00	00	01	0A	09	09	009D4	P.ABF: .LONG
00	00	00	BE	07	07	00	06	00	00	00	BD	06	06	00	009E3	.BYTE
00	00	00	0A	0A	00	08	00	00	00	BF	0B	0B	00	07	009F2	
00	C2	1B	1B	00	08	00	00	00	00	C1	0B	0B	00	0A	00	00AA01
F6	15	15	00	1C	00	00	00	00	C3	1C	1C	00	1B	00	00	00AA10
												00	15	00	00	00AA1F
																00AA24
1A	00	00	01	0D	1A	1A	00	09	00	00	01	0B	09	09	00AA28	P.ABG: .LONG
00	00	00	B2	07	07	00	06	00	00	00	B1	06	06	00	00AA37	.BYTE
00	00	00	0A	0A	00	08	00	00	00	B3	0B	0B	00	07	00AA46	
00	B7	1B	1B	00	08	00	00	00	00	B6	0B	0B	00	0A	00	00AA55
F7	15	15	00	1C	00	00	00	00	B8	1C	1C	00	1B	00	00	00AA64
												00	15	00	00	00AA73
																00AA78
08	00	00	00	41	08	08	00	07	00	00	00	40	07	07	00AA7C	P.ABH: .LONG
00	00	00	43	0B	0B	00	0A	00	00	00	42	0A	0B	00	00AA8B	.BYTE
00	00	00	1C	0B	00	1B	00	00	00	44	1B	0B	00	0B	00AA9A	
00	4A	0B	0A	00	0A	00	00	00	00	49	0A	0A	00	1C	00	00AA9A
4C	0B	0B	00	0B	00	00	00	00	4B	0A	0B	00	0B	00	00	00AAB8
1C	1C	00	1B	00	00	00	4D	1B	1B	00	0B	00	00	00	00	00AC7
												00	1C	00	00	00AD6
																00ADC
08	00	00	00	54	0E	0E	00	08	00	00	01	0B	3F	3F	00AE0	P.ABI: .LONG
00	00	00	60	07	07	00	08	00	00	00	5F	06	06	00	00AEF	.BYTE
00	00	62	0A	0A	00	08	00	00	00	61	0B	0B	00	0B	00AEF	
00	64	1B	1B	00	0B	00	00	00	00	63	0B	0B	00	0B	00	00B0D
F8	15	15	00	0B	00	00	00	65	1C	1C	00	0B	00	00	00	00B1C
												00	0B	00	00	00B2B

0. 27. 27. 9. 0. 0. 0. 27. 0. 28. 28. 10. -  
0. 0. 0. 28. 0. 21. 21. -14. 0. 0. 0. 21. -  
0. 14. 14. 82. 0. 0. 0. 14. 0  
16  
6. 6. 15. 0. 0. 0. 6. 0. 7. 7. 17. 0. 0. -  
0. 7. 0. 8. 8. 19. 0. 0. 0. 8. 0. 10. 10. -  
21. 0. 0. 0. 10. 0. 11. 11. 25. 0. 0. 0. -  
11. 0. 27. 27. 23. 0. 0. 0. 27. 0. 28. -  
28. 24. 0. 0. 0. 28. 0. 21. 21. -13. 0. -  
0. 0. 21. 0  
16  
6. 6. 43. 0. 0. 0. 6. 0. 7. 7. 45. 0. 0. -  
0. 7. 0. 8. 8. 47. 0. 0. 0. 8. 0. 10. 10. -  
46. 0. 0. 0. 10. 0. 11. 11. 50. 0. 0. 0. -  
11. 0. 27. 27. 51. 0. 0. 0. 27. 0. 28. -  
28. 52. 0. 0. 0. 28. 0. 21. 21. -12. 0. -  
0. 0. 21. 0  
16  
6. 6. 29. 0. 0. 0. 6. 0. 7. 7. 31. 0. 0. -  
0. 7. 0. 8. 8. 33. 0. 0. 0. 8. 0. 10. 10. -  
35. 0. 0. 0. 10. 0. 11. 11. 36. 0. 0. 0. -  
11. 0. 27. 27. 37. 0. 0. 0. 27. 0. 28. -  
28. 38. 0. 0. 0. 28. 0. 21. 21. -11. 0. -  
0. 0. 21. 0  
20  
9. 9. 10. 1. 0. 0. 9. 0. 26. 26. 12. 1. -  
0. 0. 26. 0. 6. 6. -67. 0. 0. 0. 6. 0. 7. -  
7. -66. 0. 0. 0. 7. 0. 8. 8. -65. 0. 0. 0. -  
0. 8. 0. 10. 10. -64. 0. 0. 0. 10. 0. 11. -  
11. -63. 0. 0. 0. 11. 0. 27. 27. -62. 0. -  
0. 0. 27. 0. 28. 28. -61. 0. 0. 0. 28. 0. -  
21. 21. -10. 0. 0. 0. 21. 0  
20  
9. 9. 11. 1. 0. 0. 9. 0. 26. 26. 13. 1. -  
0. 0. 26. 0. 6. 6. -79. 0. 0. 0. 6. 0. 7. -  
7. -78. 0. 0. 0. 7. 0. 8. 8. -77. 0. 0. 0. -  
0. 8. 0. 10. 10. -75. 0. 0. 0. 10. 0. 11. -  
11. -74. 0. 0. 0. 11. 0. 27. 27. -73. 0. -  
0. 0. 27. 0. 28. 28. -72. 0. 0. 0. 28. 0. -  
21. 21. -9. 0. 0. 0. 21. 0  
24  
7. 7. 64. 0. 0. 0. 7. 0. 8. 8. 65. 0. 0. -  
0. 8. 0. 10. 10. 66. 0. 0. 0. 10. 0. 11. -  
11. 67. 0. 0. 0. 11. 0. 27. 27. 68. 0. 0. -  
0. 27. 0. 8. 28. 69. 0. 0. 0. 28. 0. 10. -  
10. 73. 0. 0. 0. 10. 0. 11. 11. 74. 0. 0. -  
0. 11. 0. 11. 10. 75. 0. 0. 0. 11. 0. 11. -  
11. 76. 0. 0. 0. 11. 0. 27. 27. 77. 0. 0. -  
0. 27. 0. 28. 28. 78. 0. 0. 0. 28. 0  
20  
63. 63. 8. 1. 0. 0. 8. 0. 14. 14. 84. 0. -  
0. 0. 8. 0. 6. 6. 95. 0. 0. 0. 8. 0. 7. -  
7. 96. 0. 0. 0. 8. 0. 8. 8. 97. 0. 0. 0. -  
8. 0. 10. 10. 98. 0. 0. 0. 8. 0. 11. 11. -  
99. 0. 0. 0. 8. 0. 27. 27. 100. 0. 0. 0. -  
8. 0. 28. 28. 101. 0. 0. 0. 8. 0. 21. 21. -  
-8. 0. 0. 0. 8. 0

DBGEVALOP  
V04-000

M 3  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 B11gs-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32;1

Page 113  
(22)

| DBGEVALOP<br>V04-000                                     | 00000014   | 00B30  | P.ABJ: .LONG | 20  |
|--|--|--|--------------|---|
| 08 00 00 00 SE OE OE 00 08 00 00 01 09 3F 3F 00B34       | 00 00 00 6B 07 07 00 08 00 00 00 06 06 00 00B34          | 00 00 00 6C 08 08 00 08 00 00 00 08 08 00 00B43          | .BYTE        | 63, 63, 9, 1, 0, 0, 8, 0, 14, 14, 94, 0, -    |
| 00 00 6D 0A 0A 00 08 00 00 00 6C 08 08 00 00B52          | 00 6F 1B 1B 00 08 00 00 00 00 6E 08 08 00 00B61          | F9 15 15 00 08 00 00 00 70 1C 1C 00 08 00 00 00B70       |              | 0, 0, 8, 0, 10, 10, 109, 0, 0, 0, 8, 0, 11, - |
| 00 00 00 00 08 00 00 00 00 00 00 08 00 00 00B7F          | 00 00 00 00 08 00 00 00 00 00 00 08 00 00 00B84          | 00 00 00 00 08 00 00 00 00 00 00 08 00 00 00B88          | P.ABK: .LONG | 21, 21, -7, 0, 0, 0, 8, 0                     |
| 08 00 00 00 7D 06 06 00 08 00 00 00 58 0E 0E 00B88       | 00 00 00 7F 08 08 00 08 00 00 00 7E 07 07 00 00B97       | 00 00 82 0B 0B 00 08 00 00 00 81 0A 0A 00 08 00BA6       | .BYTE        | 14, 14, 88, 0, 0, 0, 8, 0, 6, 6, 125, 0, -    |
| 00 84 1C 1C 00 08 00 00 00 00 83 1B 1B 00 08 00 00B85    | 00 00 00 00 08 00 00 00 00 00 00 08 00 00 00B8C4         | 00 00 00 00 08 00 00 00 00 00 00 08 00 00 00B8D0         |              | 0, 0, 8, 0, 7, 7, 126, 0, 0, 0, 8, 0, 8, -    |
| 08 00 00 00 75 06 06 00 08 00 00 00 56 0E 0E 00BD0       | 00 00 00 77 08 08 00 08 00 00 00 76 07 07 00 00BD4       | 00 00 7A 0B 0B 00 08 00 00 00 79 0A 0A 00 08 00BE3       | P.ABL: .LONG | 8, 127, 0, 0, 0, 8, 0, 10, 10, -127, 0, -     |
| 00 00 7C 1C 00 08 00 00 00 00 00 7B 1B 1B 00 08 00 00BF2 | 00 00 00 00 08 00 00 00 00 00 00 08 00 00 00C01          | 00 00 00 00 08 00 00 00 00 00 00 08 00 00 00C10          | .BYTE        | 0, 0, 8, 0, 11, 11, 122, 0, 0, 0, 8, 0, 27, - |
| 08 00 00 00 8D 06 06 00 08 00 00 00 5C 0E 0E 00C1C       | 00 00 00 8F 08 08 00 08 00 00 00 8E 07 07 00 00C20       | 00 00 92 0B 0B 00 08 00 00 00 91 0A 0A 00 08 00C2F       | P.ABM: .LONG | 27, 123, 0, 0, 0, 8, 0, 28, 28, 124, 0, -     |
| 00 00 94 1C 1C 00 08 00 00 00 00 93 1B 1B 00 08 00 00C3E | 00 00 00 00 08 00 00 00 00 00 00 08 00 00 00C4D          | 00 00 00 00 08 00 00 00 00 00 00 08 00 00 00C5C          | .BYTE        | 0, 0, 8, 0, 21, 21, -5, 0, 0, 0, 8, 0         |
| 08 00 00 00 85 06 06 00 08 00 00 00 5A 0E 0E 00C68       | 00 00 00 87 08 08 00 08 00 00 00 86 07 07 00 00C6C       | 00 00 8A 0B 0B 00 08 00 00 00 89 0A 0A 00 08 00C7B       | P.ABN: .LONG | 14, 14, 92, 0, 0, 0, 8, 0, 6, 6, -115, 0, -   |
| 00 8C 1C 1C 00 08 00 00 00 00 00 88 1B 1B 00 08 00 00C8A | 00 00 00 00 08 00 00 00 00 00 00 88 1B 1B 00 08 00 00C99 | 00 00 00 00 08 00 00 00 00 00 00 88 1B 1B 00 08 00 00CA8 | .BYTE        | 0, 0, 8, 0, 7, 7, -114, 0, 0, 0, 8, 0, 8, -   |
| 07 00 00 00 9C 07 07 00 06 00 00 00 9B 06 06 00CB4       | 00 00 00 00 08 00 00 00 00 00 00 9B 06 06 00CB8          | 00 00 00 00 08 00 00 00 00 00 00 9B 06 06 00CC7          | P.ABO: .LONG | 8, -113, 0, 0, 0, 8, 0, 10, 10, -111, 0, -    |
| 07 00 00 00 96 07 07 00 06 00 00 00 95 06 06 00CD0       | 00 00 00 00 08 00 00 00 00 00 00 95 06 06 00CD4          | 00 00 00 00 08 00 00 00 00 00 00 95 06 06 00CE3          | P.ABP: .LONG | 0, 0, 8, 0, 11, 11, -110, 0, 0, 0, 8, 0, -    |
| 07 00 00 00 9F 07 07 00 06 00 00 00 9E 06 06 00CEC       | 00 00 00 00 08 00 00 00 00 00 00 9E 06 06 00CF0          | 00 00 00 00 08 00 00 00 00 00 00 9E 06 06 00CFF          | .BYTE        | 27, 123, 0, 0, 0, 8, 0, 28, 28, 124, 0, -     |
| 07 00 00 00 A2 07 07 00 06 00 00 00 A1 06 06 00D08       | 00 00 00 00 08 00 00 00 00 00 00 A1 06 06 00D0C          | 00 00 00 00 08 00 00 00 00 00 00 A1 06 06 00D1B          | P.ABQ: .LONG | -108, 0, 0, 0, 8, 0, 21, 21, -4, 0, 0, 0, -   |
| 07 00 00 00 99 07 07 00 06 00 00 00 98 06 06 00D24       | 00 00 00 00 08 00 00 00 00 00 00 98 06 06 00D28          | 00 00 00 00 08 00 00 00 00 00 00 98 06 06 00D37          | P.ABR: .LONG | 8, 0  |
| 07 00 00 00 E0 07 07 00 06 00 00 00 DF 06 06 00D40       | 00 00 00 00 08 00 00 00 00 00 00 DF 06 06 00D44          | 00 00 00 00 08 00 00 00 00 00 00 DF 06 06 00D53          | P.ABS: .LONG | 14, 14, 90, 0, 0, 0, 8, 0, 6, 6, -123, 0, -   |
|  |  |  | .BYTE        | 0, 0, 8, 0, 7, 7, -122, 0, 0, 0, 8, 0, 8, -   |
|  |  |  |              | 8, -121, 0, 0, 0, 8, 0, 10, 10, -119, 0, -    |
|  |  |  |              | 0, 0, 8, 0, 11, 11, -118, 0, 0, 0, 8, 0, -    |
|  |  |  |              | 27, 27, -117, 0, 0, 0, 8, 0, 28, 28, -        |
|  |  |  |              | -116, 0, 0, 0, 8, 0, 21, 21, -3, 0, 0, 0, -   |
|  |  |  |              | 8, 0  |
|  |  |  |              | 6, 6, -101, 0, 0, 0, 6, 0, 7, 7, -100, 0, -   |
|  |  |  |              | 0, 0, 7, 0, 8, 8, -99, 0, 0, 0, 8, 0          |
|  |  |  |              | 6, 6, -107, 0, 0, 0, 6, 0, 7, 7, -106, 0, -   |
|  |  |  |              | 0, 0, 7, 0, 8, 8, -105, 0, 0, 0, 8, 0         |
|  |  |  |              | 6, 6, -98, 0, 0, 0, 6, 0, 7, 7, -97, 0, -     |
|  |  |  |              | 0, 0, 7, 0, 8, 8, -96, 0, 0, 0, 8, 0          |
|  |  |  |              | 6, 6, -95, 0, 0, 0, 6, 0, 7, 7, -94, 0, -     |
|  |  |  |              | 0, 0, 7, 0, 8, 8, -93, 0, 0, 0, 8, 0          |
|  |  |  |              | 6, 6, -104, 0, 0, 0, 6, 0, 7, 7, -103, 0, -   |
|  |  |  |              | 0, 0, 7, 0, 8, 8, -102, 0, 0, 0, 8, 0         |
|  |  |  |              | 6, 6, -33, 0, 0, 0, 6, 0, 7, 7, -32, 0, -     |
|  |  |  |              | 0, 0, 7, 0, 8, 8, -31, 0, 0, 0, 8, 0          |

| 00# 00D5C BASIC_OPINFO TABLE: |          |          |       |
|-------------------------------|----------|----------|-------|
| 00000000                      | 00000880 | 000009D4 | 00D9C |
|                               |          | 01       | 00DA8 |
|                               |          | 00#      | 00DA9 |
| 00000000                      | 00000880 | 00000A28 | 00DAC |
|                               |          | 01       | 00DB8 |
|                               |          | 00#      | 00DB9 |
| 00000000                      | 00000880 | 000008BC | 00DBC |
|                               |          | 01       | 00DC8 |
|                               |          | 00#      | 00DC9 |
| 00000000                      | 00000880 | 00000908 | 00DCC |
|                               |          | 01       | 00DD8 |
|                               |          | 00#      | 00DD9 |
| 00000000                      | 00000880 | 0000094C | 00DDC |
|                               |          | 01       | 00DE8 |
|                               |          | 00#      | 00DE9 |
| 00000000                      | 00000880 | 00000990 | 00DEC |
|                               |          | 01       | 00DF8 |
|                               |          | 00#      | 00DF9 |
| 00000000                      | 00000880 | 00000A7C | 00DFC |
|                               |          | 01       | 00E08 |
|                               |          | 00#      | 00E09 |
| 00000000                      | 00000880 | 00000AE0 | 00E2C |
|                               |          | 01       | 00E38 |
|                               |          | 00#      | 00E39 |
| 00000000                      | 00000880 | 00000B34 | 00E3C |
|                               |          | 01       | 00E48 |
|                               |          | 00#      | 00E49 |
| 00000000                      | 00000880 | 00000B88 | 00E4C |
|                               |          | 01       | 00E58 |
|                               |          | 00#      | 00E59 |
| 00000000                      | 00000880 | 00000BD4 | 00E6C |
|                               |          | 01       | 00E78 |
|                               |          | 00#      | 00E79 |
| 00000000                      | 00000880 | 00000C20 | 00E8C |
|                               |          | 01       | 00E98 |
|                               |          | 00#      | 00E99 |
| 00000000                      | 00000880 | 00000C6C | 00EAC |
|                               |          | 01       | 00EB8 |
|                               |          | 00#      | 00EB9 |
| 00000000                      | 00000898 | 00000CB8 | 00F3C |
|                               |          | 01       | 00F48 |
|                               |          | 00#      | 00F49 |
| 00000000                      | 00000898 | 00000CD4 | 00F4C |
|                               |          | 01       | 00F58 |
|                               |          | 00#      | 00F59 |
| 00000000                      | 00000898 | 00000CF0 | 00F5C |
|                               |          | 01       | 00F68 |
|                               |          | 00#      | 00F69 |
| 00000000                      | 00000898 | 00000D0C | 00F6C |
|                               |          | 01       | 00F78 |
|                               |          | 00#      | 00F79 |
| 00000000                      | 00000898 | 00000D28 | 00F7C |
|                               |          | 01       | 00F88 |
|                               |          | 00#      | 00F89 |
| 00000000                      | 000008A4 | 00000000 | 0107C |

.....

|      |      |      |      |      |      |      |      |      |          |          |              |  |       |               |       |              |                                |              |                        |
|------|------|------|------|------|------|------|------|------|----------|----------|--------------|--|-------|---------------|-------|--------------|--------------------------------|--------------|------------------------|
|      |      |      |      |      |      |      |      |      | 01       | 01088    | .BYTE        | 1  |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00#      | 01089    | .BYTE        | 0[3]                                       |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00000000 | 00000898 | 00000000     | 0108C                                      | .LONG | 0, 2200, 0    |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 01       | 01098    | .BYTE        | 1  |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00#      | 01099    | .BYTE        | 0[67]                                      |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00000000 | 00000000 | 00000000     | 010DC                                      | .LONG | 0, 0, 0       |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 01       | 010E8    | .BYTE        | 1  |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00#      | 010E9    | .BYTE        | 0[19]                                      |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00000000 | 00000898 | 00000D44     | 010FC                                      | .LONG | 3396, 2200, 0 |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 01       | 01108    | .BYTE        | 1  |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      |          | 01109    | .BLKB        | 147  |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00000001 | 0119C    | .LONG        | 1  |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 083B     | 011A0    | P.ABU: .WORD | 2107                                       |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      |          | 011A2    | .BLKB        | 2  |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00000000 | 000011A0 | 011A4        | BLISS_CVTINFO_TABLE:                       |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      |          |          | .LONG        | 4512, 0                                    |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 011AC    | .BYTE        | 0  |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      |          | 011AD    | .BLKB        | 0  |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00000003 | 011B0    | .LONG        | 0  |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 0000     | 0807     | 0706         | 0804                                       | 0403  | 0302          | 011B4 | P.ABV: .WORD | 770, 1027, 2052, 1798, 2055, 0 |              |                        |
|      |      |      |      |      |      |      |      |      | 00000006 | 011C0    | .LONG        | 6  |       |               |       |              |                                |              |                        |
| 0601 | 0129 | 292A | 2A22 | 2204 | 0408 | 0803 | 0307 | 0702 | 0206     | 011C4    | P.ABW: .WORD | 518, 1794, 775, 2051, 1032, 8708, 10786, - |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 0000     | 011D8    | .WORD        | 10538, 297, 1537, 0                        |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      |          | 011DA    | .BLKB        | 2  |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00000002 | 011DC    | .LONG        | 2  |       |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 05            | 08    | 08           | 011E0                          | P.ABX: .BYTE | 8, 5, 0, 0, 0, 8, 0    |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 011E8        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 13            | 08    | 08           | 011EC                          | P.ABY: .BYTE | 8, 19, 0, 0, 0, 8, 0   |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 011F4        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 2F            | 08    | 08           | 011F8                          | P.ABZ: .BYTE | 8, 47, 0, 0, 0, 8, 0   |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 01200        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 21            | 08    | 08           | 01204                          | P.ACA: .BYTE | 8, 33, 0, 0, 0, 8, 0   |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 0120C        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 3B            | 08    | 08           | 01210                          | P.ACB: .BYTE | 8, 59, 0, 0, 0, 8, 0   |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 01218        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 3D            | 08    | 08           | 0121C                          | P.ACC: .BYTE | 8, 61, 0, 0, 0, 8, 0   |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 01224        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 61            | 08    | 08           | 01228                          | P.ACD: .BYTE | 8, 97, 0, 0, 0, 8, 0   |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 01230        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 6C            | 08    | 08           | 01234                          | P.ACE: .BYTE | 8, 108, 0, 0, 0, 8, 0  |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 0123C        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 8F            | 08    | 08           | 01240                          | P.ACF: .BYTE | 8, -113, 0, 0, 0, 8, 0 |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 01248        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 90            | 08    | 08           | 0124C                          | P.ACG: .BYTE | 8, -112, 0, 0, 0, 8, 0 |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 01254        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 7F            | 08    | 08           | 01258                          | P.ACH: .BYTE | 8, 127, 0, 0, 0, 8, 0  |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 01260        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 80            | 08    | 08           | 01264                          | P.ACI: .BYTE | 8, -128, 0, 0, 0, 8, 0 |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 0126C        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 87            | 08    | 08           | 01270                          | P.ACJ: .BYTE | 8, -121, 0, 0, 0, 8, 0 |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 01278        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 88            | 08    | 08           | 0127C                          | P.ACK: .BYTE | 8, -120, 0, 0, 0, 8, 0 |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 01284        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 77            | 08    | 08           | 01288                          | P.ACL: .BYTE | 8, 119, 0, 0, 0, 8, 0  |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 01290        | .LONG                                      | 2     |               |       |              |                                |              |                        |
|      |      |      |      |      |      |      |      |      | 00       | 08       | 00           | 00   | 00    | 78            | 08    | 08           | 01294                          | P.ACM: .BYTE | 8, 120, 0, 0, 0, 8, 0  |
|      |      |      |      |      |      |      |      |      |          | 00000002 | 0129C        | .LONG                                      | 2     |               |       |              |                                |              |                        |

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        |       |  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|--------|-------|--|
| 07 | 00 | 00 | 00 | BE | 07 | 07 | 00 | 06 | 00 | 00 | 00 | BD | 06 | 06 | 012A0 | P.ACN: | .BYTE | 8, 8, -105, 0, 0, 0, 8, 0                      |
| 00 | 00 | 00 | C1 | 0B | 0B | 00 | 0A | 00 | 00 | 00 | C0 | 0A | 0A | 00 | 012A8 |        | .LONG | 2  |
| 00 | 00 | C3 | 1C | 1C | 00 | 1B | 00 | 00 | 00 | C2 | 1B | 1B | 00 | 0B | 012AC | P.ACO: | .BYTE | 8, 8, -96, 0, 0, 0, 8, 0                       |
| 01 | 0A | 09 | 09 | 00 | 15 | 00 | 00 | 00 | F6 | 15 | 15 | 00 | 1C | 00 | 012B4 |        | .LONG | 2  |
| BF | 0B | 0B | 00 | 1A | 00 | 00 | 01 | 0C | 1A | 1A | 00 | 09 | 00 | 00 | 012B8 | P.ACP: | .BYTE | 8, 8, -93, 0, 0, 0, 8, 0                       |
|    |    |    |    |    |    |    |    |    |    |    | 00 | 08 | 00 | 00 | 012C0 |        | .LONG | 2  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 012C4 | P.ACQ: | .BYTE | 8, 8, -102, 0, 0, 0, 8, 0                      |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 012CC |        | .LONG | 20   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 012D0 | P.ACR: | .BYTE | 6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -      |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 012DF |        | .LONG | 0, 0, 7, 0, 10, 10, -64, 0, 0, 0, 10, 0, -     |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 012EE |        | .LONG | 11, 11, -63, 0, 0, 0, 11, 0, 27, 27, -62, 0, - |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 012FD |        | .LONG | 0, 0, 0, 27, 0, 28, 28, -61, 0, 0, 0, 28, -    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0130C |        | .LONG | 0, 21, 21, -10, 0, 0, 0, 21, 0, 9, 9, 10, -    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0131B |        | .LONG | 1, 0, 0, 0, 0, 26, 26, 12, 1, 0, 0, 26, -      |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | .LONG | 0, 8, 8, -65, 0, 0, 0, 8, 0                    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 01320 | P.ACS: | .LONG | 20   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 01324 |        | .BYTE | 6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -      |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 01333 |        | .LONG | 0, 0, 7, 0, 10, 10, -75, 0, 0, 0, 10, 0, -     |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 01342 |        | .LONG | 11, 11, -74, 0, 0, 0, 11, 0, 27, 27, -73, 0, - |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 01351 |        | .LONG | 0, 0, 0, 27, 0, 28, 28, -72, 0, 0, 0, 28, -    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 01360 |        | .LONG | 0, 21, 21, -9, 0, 0, 0, 21, 0, 9, 9, 11, -     |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0136F |        | .LONG | 1, 0, 0, 0, 0, 26, 26, 13, 1, 0, 0, 26, -      |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | .LONG | 0, 8, 8, -77, 0, 0, 0, 8, 0                    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 01374 | P.ACT: | .LONG | 2  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 01378 |        | .BYTE | 8, 8, -99, 0, 0, 0, 8, 0                       |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 01380 |        | .LONG | 2  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 01384 | P.ACU: | .BYTE | 8, 8, -47, 0, 0, 0, 8, 0                       |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0138C |        | .LONG | 20   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 01390 | P.ACV: | .BYTE | 6, 6, -49, 0, 0, 0, 6, 0, 7, 7, -49, 0, -      |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0139F |        | .LONG | 0, 0, 8, 0, 2, 2, -49, 0, 0, 0, 8, 0, 3, -     |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 013AE |        | .LONG | 3, -49, 0, 0, 0, 8, 0, 4, 4, -49, 0, 0, -      |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 013BD |        | .LONG | 0, 8, 0, 1, 1, -49, 0, 0, 0, 8, 0, 34, -       |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 013CC |        | .LONG | 34, -49, 0, 0, 0, 8, 0, 41, 41, -49, 0, -      |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 013DB |        | .LONG | 0, 0, 8, 0, 42, 42, -49, 0, 0, 0, 8, 0, -      |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | .LONG | 8, 8, -49, 0, 0, 0, 8, 0                       |

## 00# 013E0 BLISS\_OPINFO TABLE:

|          |          |          |       |       |               |
|----------|----------|----------|-------|-------|---------------|
| 00000000 | 000011B4 | 00001390 | 01410 | .BYTE | 0[48]         |
|          |          | 01       | 0141C | .LONG | 5008, 4532, 0 |
|          |          | 00#      | 0141D | .BYTE | 1             |
| 00000000 | 000011B4 | 000012D0 | 01420 | .BYTE | 0[3]          |
|          |          | 00       | 0142C | .LONG | 4816, 4532, 0 |
|          |          | 00#      | 0142D | .BYTE | 0             |
| 00000000 | 000011B4 | 00001324 | 01430 | .BYTE | 0[3]          |
|          |          | 00       | 0143C | .LONG | 4900, 4532, 0 |
|          |          | 00#      | 0143D | .BYTE | 0             |
| 00000000 | 000011B4 | 000011E0 | 01440 | .BYTE | 0[3]          |
|          |          | 00       | 0144C | .LONG | 4576, 4532, 0 |
|          |          | 00#      | 0144D | .BYTE | 0             |
| 00000000 | 000011B4 | 000011EC | 01450 | .BYTE | 0[3]          |
|          |          | 00       | 0145C | .LONG | 4588, 4532, 0 |
|          |          | 00#      | 0145D | .BYTE | 0             |
| 00000000 | 000011B4 | 000011F8 | 01460 | .BYTE | 0[3]          |
|          |          | 00       | 0146C | .LONG | 4600, 4532, 0 |
|          |          | 00#      | 0146D | .BYTE | 0             |
| 00000000 | 000011B4 | 00001204 | 01470 | .BYTE | 0[3]          |
|          |          | 00       | 0147C | .LONG | 4612, 4532, 0 |
|          |          |          |       | .BYTE | 0             |

|          |          |           |           |        |       |       |   |
|----------|----------|-----------|-----------|--------|-------|-------|---|
| 00000000 | 000011B4 | 00001228  | 00# 0147D | .BYTE  | 0[51] |       |   |
|          |          | 00# 01480 | .LONG     | 4648,  | 4532, | 0     |   |
|          |          | 00# 014BC | .BYTE     | 0      |       |       |   |
|          |          | 00# 014BD | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 00001234  | 00# 014C0 | .LONG  | 4660, | 4532, | 0 |
|          |          | 00# 014CC | .BYTE     | 0      |       |       |   |
|          |          | 00# 014CD | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 00001258  | 00# 014D0 | .LONG  | 4696, | 4532, | 0 |
|          |          | 00# 014DC | .BYTE     | 0      |       |       |   |
|          |          | 00# 014DD | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 00001264  | 00# 014E0 | .LONG  | 4708, | 4532, | 0 |
|          |          | 00# 014EC | .BYTE     | 0      |       |       |   |
|          |          | 00# 014ED | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 00001288  | 00# 014F0 | .LONG  | 4744, | 4532, | 0 |
|          |          | 00# 014FC | .BYTE     | 0      |       |       |   |
|          |          | 00# 014FD | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 00001294  | 00# 01500 | .LONG  | 4756, | 4532, | 0 |
|          |          | 00# 0150C | .BYTE     | 0      |       |       |   |
|          |          | 00# 0150D | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 00001240  | 00# 01510 | .LONG  | 4672, | 4532, | 0 |
|          |          | 00# 0151C | .BYTE     | 0      |       |       |   |
|          |          | 00# 0151D | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 0000124C  | 00# 01520 | .LONG  | 4684, | 4532, | 0 |
|          |          | 00# 0152C | .BYTE     | 0      |       |       |   |
|          |          | 00# 0152D | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 00001270  | 00# 01530 | .LONG  | 4720, | 4532, | 0 |
|          |          | 00# 0153C | .BYTE     | 0      |       |       |   |
|          |          | 00# 0153D | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 0000127C  | 00# 01540 | .LONG  | 4732, | 4532, | 0 |
|          |          | 00# 0154C | .BYTE     | 0      |       |       |   |
|          |          | 00# 0154D | .BYTE     | 0[115] |       |       |   |
| 00000000 | 000011B4 | 00001378  | 00# 015C0 | .LONG  | 4984, | 4532, | 0 |
|          |          | 00# 015CC | .BYTE     | 0      |       |       |   |
|          |          | 00# 015CD | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 000012A0  | 00# 015D0 | .LONG  | 4768, | 4532, | 0 |
|          |          | 00# 015DC | .BYTE     | 0      |       |       |   |
|          |          | 00# 015DD | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 000012AC  | 00# 015E0 | .LONG  | 4780, | 4532, | 0 |
|          |          | 00# 015EC | .BYTE     | 0      |       |       |   |
|          |          | 00# 015ED | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 000012B8  | 00# 015F0 | .LONG  | 4792, | 4532, | 0 |
|          |          | 00# 015FC | .BYTE     | 0      |       |       |   |
|          |          | 00# 015FD | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 000012C4  | 00# 01600 | .LONG  | 4804, | 4532, | 0 |
|          |          | 00# 0160C | .BYTE     | 0      |       |       |   |
|          |          | 00# 0160D | .BYTE     | 0[35]  |       |       |   |
| 00000000 | 000011B4 | 00001210  | 00# 01630 | .LONG  | 4624, | 4532, | 0 |
|          |          | 00# 0163C | .BYTE     | 0      |       |       |   |
|          |          | 00# 0163D | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011B4 | 0000121C  | 00# 01640 | .LONG  | 4636, | 4532, | 0 |
|          |          | 00# 0164C | .BYTE     | 0      |       |       |   |
|          |          | 00# 0164D | .BYTE     | 0[163] |       |       |   |
| 00000000 | 000011B4 | 00001384  | 00# 016F0 | .LONG  | 4996, | 4532, | 0 |
|          |          | 00# 016FC | .BYTE     | 0      |       |       |   |
|          |          | 00# 016FD | .BYTE     | 0[3]   |       |       |   |
| 00000000 | 000011C4 | 00000000  | 00# 01700 | .LONG  | 0,    | 4548, | 0 |
|          |          | 00# 0170C | .BYTE     | 0      |       |       |   |

| Address | Op | Op2 | Op3 | Op4 | Op5 | Op6 | Op7 | Op8 | Op9 | Op10 | Op11 | Op12 | Op13 | Op14 | Op15 | Op16 | Op17 | Op18 | Op19 | Op20 | Op21 | Op22 | Op23 | Op24 | Op25 | Op26 | Op27 | Op28 | Op29 | Op30 | Op31 | Op32 | Op33 | Op34 | Op35 | Op36 | Op37 | Op38 | Op39 | Op40 | Op41 | Op42 | Op43 | Op44 | Op45 | Op46 | Op47 | Op48 | Op49 | Op50 | Op51 | Op52 | Op53 | Op54 | Op55 | Op56 | Op57 | Op58 | Op59 | Op60 | Op61 | Op62 | Op63 | Op64 | Op65 | Op66 | Op67 | Op68 | Op69 | Op70 | Op71 | Op72 | Op73 | Op74 | Op75 | Op76 | Op77 | Op78 | Op79 | Op80 | Op81 | Op82 | Op83 | Op84 | Op85 | Op86 | Op87 | Op88 | Op89 | Op90 | Op91 | Op92 | Op93 | Op94 | Op95 | Op96 | Op97 | Op98 | Op99 | Op100 | Op101 | Op102 | Op103 | Op104 | Op105 | Op106 | Op107 | Op108 | Op109 | Op110 | Op111 | Op112 | Op113 | Op114 | Op115 | Op116 | Op117 | Op118 | Op119 | Op120 | Op121 | Op122 | Op123 | Op124 | Op125 | Op126 | Op127 | Op128 | Op129 | Op130 | Op131 | Op132 | Op133 | Op134 | Op135 | Op136 | Op137 | Op138 | Op139 | Op140 | Op141 | Op142 | Op143 | Op144 | Op145 | Op146 | Op147 | Op148 | Op149 | Op150 | Op151 | Op152 | Op153 | Op154 | Op155 | Op156 | Op157 | Op158 | Op159 | Op160 | Op161 | Op162 | Op163 | Op164 | Op165 | Op166 | Op167 | Op168 | Op169 | Op170 | Op171 | Op172 | Op173 | Op174 | Op175 | Op176 | Op177 | Op178 | Op179 | Op180 | Op181 | Op182 | Op183 | Op184 | Op185 | Op186 | Op187 | Op188 | Op189 | Op190 | Op191 | Op192 | Op193 | Op194 | Op195 | Op196 | Op197 | Op198 | Op199 | Op200 | Op201 | Op202 | Op203 | Op204 | Op205 | Op206 | Op207 | Op208 | Op209 | Op210 | Op211 | Op212 | Op213 | Op214 | Op215 | Op216 | Op217 | Op218 | Op219 | Op220 | Op221 | Op222 | Op223 | Op224 | Op225 | Op226 | Op227 | Op228 | Op229 | Op230 | Op231 | Op232 | Op233 | Op234 | Op235 | Op236 | Op237 | Op238 | Op239 | Op240 | Op241 | Op242 | Op243 | Op244 | Op245 | Op246 | Op247 | Op248 | Op249 | Op250 | Op251 | Op252 | Op253 | Op254 | Op255 | Op256 | Op257 | Op258 | Op259 | Op260 | Op261 | Op262 | Op263 | Op264 | Op265 | Op266 | Op267 | Op268 | Op269 | Op270 | Op271 | Op272 | Op273 | Op274 | Op275 | Op276 | Op277 | Op278 | Op279 | Op280 | Op281 | Op282 | Op283 | Op284 | Op285 | Op286 | Op287 | Op288 | Op289 | Op290 | Op291 | Op292 | Op293 | Op294 | Op295 | Op296 | Op297 | Op298 | Op299 | Op300 | Op301 | Op302 | Op303 | Op304 | Op305 | Op306 | Op307 | Op308 | Op309 | Op310 | Op311 | Op312 | Op313 | Op314 | Op315 | Op316 | Op317 | Op318 | Op319 | Op320 | Op321 | Op322 | Op323 | Op324 | Op325 | Op326 | Op327 | Op328 | Op329 | Op330 | Op331 | Op332 | Op333 | Op334 | Op335 | Op336 | Op337 | Op338 | Op339 | Op340 | Op341 | Op342 | Op343 | Op344 | Op345 | Op346 | Op347 | Op348 | Op349 | Op350 | Op351 | Op352 | Op353 | Op354 | Op355 | Op356 | Op357 | Op358 | Op359 | Op360 | Op361 | Op362 | Op363 | Op364 | Op365 | Op366 | Op367 | Op368 | Op369 | Op370 | Op371 | Op372 | Op373 | Op374 | Op375 | Op376 | Op377 | Op378 | Op379 | Op380 | Op381 | Op382 | Op383 | Op384 | Op385 | Op386 | Op387 | Op388 | Op389 | Op390 | Op391 | Op392 | Op393 | Op394 | Op395 | Op396 | Op397 | Op398 | Op399 | Op400 | Op401 | Op402 | Op403 | Op404 | Op405 | Op406 | Op407 | Op408 | Op409 | Op410 | Op411 | Op412 | Op413 | Op414 | Op415 | Op416 | Op417 | Op418 | Op419 | Op420 | Op421 | Op422 | Op423 | Op424 | Op425 | Op426 | Op427 | Op428 | Op429 | Op430 | Op431 | Op432 | Op433 | Op434 | Op435 | Op436 | Op437 | Op438 | Op439 | Op440 | Op441 | Op442 | Op443 | Op444 | Op445 | Op446 | Op447 | Op448 | Op449 | Op450 | Op451 | Op452 | Op453 | Op454 | Op455 | Op456 | Op457 | Op458 | Op459 | Op460 | Op461 | Op462 | Op463 | Op464 | Op465 | Op466 | Op467 | Op468 | Op469 | Op470 | Op471 | Op472 | Op473 | Op474 | Op475 | Op476 | Op477 | Op478 | Op479 | Op480 | Op481 | Op482 | Op483 | Op484 | Op485 | Op486 | Op487 | Op488 | Op489 | Op490 | Op491 | Op492 | Op493 | Op494 | Op495 | Op496 | Op497 | Op498 | Op499 | Op500 | Op501 | Op502 | Op503 | Op504 | Op505 | Op506 | Op507 | Op508 | Op509 | Op510 | Op511 | Op512 | Op513 | Op514 | Op515 | Op516 | Op517 | Op518 | Op519 | Op520 | Op521 | Op522 | Op523 | Op524 | Op525 | Op526 | Op527 | Op528 | Op529 | Op530 | Op531 | Op532 | Op533 | Op534 | Op535 | Op536 | Op537 | Op538 | Op539 | Op540 | Op541 | Op542 | Op543 | Op544 | Op545 | Op546 | Op547 | Op548 | Op549 | Op550 | Op551 | Op552 | Op553 | Op554 | Op555 | Op556 | Op557 | Op558 | Op559 | Op560 | Op561 | Op562 | Op563 | Op564 | Op565 | Op566 | Op567 | Op568 | Op569 | Op570 | Op571 | Op572 | Op573 | Op574 | Op575 | Op576 | Op577 | Op578 | Op579 | Op580 | Op581 | Op582 | Op583 | Op584 | Op585 | Op586 | Op587 | Op588 | Op589 | Op590 | Op591 | Op592 | Op593 | Op594 | Op595 | Op596 | Op597 | Op598 | Op599 | Op600 | Op601 | Op602 | Op603 | Op604 | Op605 | Op606 | Op607 | Op608 | Op609 | Op610 | Op611 | Op612 | Op613 | Op614 | Op615 | Op616 | Op617 | Op618 | Op619 | Op620 | Op621 | Op622 | Op623 | Op624 | Op625 | Op626 | Op627 | Op628 | Op629 | Op630 | Op631 | Op632 | Op633 | Op634 | Op635 | Op636 | Op637 | Op638 | Op639 | Op640 | Op641 | Op642 | Op643 | Op644 | Op645 | Op646 | Op647 | Op648 | Op649 | Op650 | Op651 | Op652 | Op653 | Op654 | Op655 | Op656 | Op657 | Op658 | Op659 | Op660 | Op661 | Op662 | Op663 | Op664 | Op665 | Op666 | Op667 | Op668 | Op669 | Op670 | Op671 | Op672 | Op673 | Op674 | Op675 | Op676 | Op677 | Op678 | Op679 | Op680 | Op681 | Op682 | Op683 | Op684 | Op685 | Op686 | Op687 | Op688 | Op689 | Op690 | Op691 | Op692 | Op693 | Op694 | Op695 | Op696 | Op697 | Op698 | Op699 | Op700 | Op701 | Op702 | Op703 | Op704 | Op705 | Op706 | Op707 | Op708 | Op709 | Op710 | Op711 | Op712 | Op713 | Op714 | Op715 | Op716 | Op717 | Op718 | Op719 | Op720 | Op721 | Op722 | Op723 | Op724 | Op725 | Op726 | Op727 | Op728 | Op729 | Op730 | Op731 | Op732 | Op733 | Op734 | Op735 | Op736 | Op737 | Op738 | Op739 | Op740 | Op741 | Op742 | Op743 | Op744 | Op745 | Op746 | Op747 | Op748 | Op749 | Op750 | Op751 | Op752 | Op753 | Op754 | Op755 | Op756 | Op757 | Op758 | Op759 | Op760 | Op761 | Op762 | Op763 | Op764 | Op765 | Op766 | Op767 | Op768 | Op769 | Op770 | Op771 | Op772 | Op773 | Op774 | Op775 | Op776 | Op777 | Op778 | Op779 | Op780 | Op781 | Op782 | Op783 | Op784 | Op785 | Op786 | Op787 | Op788 | Op789 | Op790 | Op791 | Op792 | Op793 | Op794 | Op795 | Op796 | Op797 | Op798 | Op799 | Op800 | Op801 | Op802 | Op803 | Op804 | Op805 | Op806 | Op807 | Op808 | Op809 | Op810 | Op811 | Op812 | Op813 | Op814 | Op815 | Op816 | Op817 | Op818 | Op819 | Op820 | Op821 | Op822 | Op823 | Op824 | Op825 | Op826 | Op827 | Op828 | Op829 | Op830 | Op831 | Op832 | Op833 | Op834 | Op835 | Op836 | Op837 | Op838 | Op839 | Op840 | Op841 | Op842 | Op843 | Op844 | Op845 | Op846 | Op847 | Op848 | Op849 | Op850 | Op851 | Op852 | Op853 | Op854 | Op855 | Op856 | Op857 | Op858 | Op859 | Op860 | Op861 | Op862 | Op863 | Op864 | Op865 | Op866 | Op867 | Op868 | Op869 | Op870 | Op871 | Op872 | Op873 | Op874 | Op875 | Op876 | Op877 | Op878 | Op879 | Op880 | Op881 | Op882 | Op883 | Op884 | Op885 | Op886 | Op887 | Op888 | Op889 | Op890 | Op891 | Op892 | Op893 | Op894 | Op895 | Op896 | Op897 | Op898 | Op899 | Op900 | Op901 | Op902 | Op903 | Op904 | Op905 | Op906 | Op907 | Op908 | Op909 | Op910 | Op911 | Op912 | Op913 | Op914 | Op915 | Op916 | Op917 | Op918 | Op919 | Op920 | Op921 | Op922 | Op923 | Op924 | Op925 | Op926 | Op927 | Op928 | Op929 | Op930 | Op931 | Op932 | Op933 | Op934 | Op935 | Op936 | Op937 | Op938 | Op939 | Op940 | Op941 | Op942 | Op943 | Op944 | Op945 | Op946 | Op947 | Op948 | Op949 | Op950 | Op951 | Op952 | Op953 | Op954 | Op955 | Op956 | Op957 | Op958 | Op959 | Op960 | Op961 | Op962 | Op963 | Op964 | Op965 | Op966 | Op967 | Op968 | Op969 | Op970 | Op971 | Op972 | Op973 | Op974 | Op975 | Op976 | Op977 | Op978 | Op979 | Op980 | Op981 | Op982 | Op983 | Op984 | Op985 | Op986 | Op987 | Op988 | Op989 | Op990 | Op991 | Op992 | Op993 | Op994 | Op995 | Op996 | Op997 | Op998 | Op999 | Op1000 | Op1001 | Op1002 | Op1003 | Op1004 | Op1005 | Op1006 | Op1007 | Op1008 | Op1009 | Op1010 | Op1011 | Op1012 | Op1013 | Op1014 | Op1015 | Op1016 | Op1017 | Op1018 | Op1019 | Op1020 | Op1021 | Op1022 | Op1023 | Op1024 | Op1025 | Op1026 | Op1027 | Op1028 | Op1029 | Op1030 | Op1031 | Op1032 | Op1033 | Op1034 | Op1035 | Op1036 | Op1037 | Op1038 | Op1039 | Op1040 | Op1041 | Op1042 | Op1043 | Op1044 | Op1045 | Op1046 | Op1047 | Op1048 | Op1049 | Op1050 | Op1051 | Op1052 | Op1053 | Op1054 | Op1055 | Op1056 | Op1057 | Op1058 | Op1059 | Op1060 | Op1061 | Op1062 | Op1063 | Op1064 | Op1065 | Op1066 | Op1067 | Op1068 | Op1069 | Op1070 | Op1071 | Op1072 | Op1073 | Op1074 | Op1075 | Op1076 | Op1077 | Op1078 | Op1079 | Op1080 | Op1081 | Op1082 | Op1083 | Op1084 | Op1085 | Op1086 | Op1087 | Op1088 | Op1089 | Op1090 | Op1091 | Op1092 | Op1093 | Op1094 | Op1095 | Op1096 | Op1097 | Op1098 | Op1099 | Op1100 | Op1101 | Op1102 | Op1103 | Op1104 | Op1105 | Op1106 | Op1107 | Op1108 | Op1109 | Op1110 | Op1111 | Op1112 | Op1113 | Op1114 | Op1115 | Op1116 | Op1117 | Op1118 | Op1119 | Op1120 | Op1121 | Op1122 | Op1123 | Op1124 | Op1125 | Op1126 | Op1127 | Op1128 | Op1129 | Op1130 | Op1131 | Op1132 | Op1133 | Op1134 | Op1135 | Op1136 | Op1137 | Op1138 | Op1139 | Op1140 | Op1141 | Op1142 | Op1143 | Op1144 | Op1145 | Op1146 | Op1147 | Op1148 | Op1149 | Op1150 | Op1151 | Op1152 | Op1153 | Op1154 | Op1155 | Op1156 | Op1157 | Op1158 | Op1159 | Op1160 | Op1161 | Op1162 | Op1163 | Op1164 | Op1165 | Op1166 | Op1167 | Op1168 | Op1169 | Op1170 | Op1171 | Op1172 | Op1173 | Op1174 | Op1175 | Op1176 | Op1177 | Op1178 | Op1179 | Op1180 | Op1181 | Op1182 | Op1183 | Op1184 | Op1185 | Op1186 | Op1187 | Op1188 | Op1189 | Op1190 | Op1191 | Op1192 | Op1193 | Op1194 | Op1195 | Op1196 | Op1197 | Op1198 | Op1199 | Op1200 | Op1201 | Op1202 | Op1203 | Op1204 | Op1205 | Op1206 | Op1207 | Op1208 | Op1209 | Op1210 | Op1211 | Op1212 | Op1213 | Op1214 | Op1215 | Op1216 | Op1217 | Op1218 | Op1219 | Op1220 | Op1221 | Op1222 | Op1223 | Op1224 | Op1225 | Op1226 | Op1227 | Op1228 | Op1229 | Op1230 | Op1231 | Op1232 | Op1233 | Op1234 | Op1235 | Op1236 | Op1237 | Op1238 | Op1239 | Op1240 | Op1241 | Op1242 | Op1243 | Op1244 | Op1245 | Op1246 | Op1247 | Op1248 | Op1249 | Op1250 | Op1251 | Op1252 | Op1253 | Op1254 | Op1255 | Op1256 | Op1257 | Op1258 | Op1259 | Op1260 | Op1261 | Op1262 | Op1263 | Op1264 | Op1265 | Op1266 | Op1267 | Op1268 | Op1269 | Op1270 | Op1271 | Op1272 | Op1273 | Op1274 | Op1275 | Op1276 | Op1277 | Op1278 | Op1279 | Op1280 | Op1281 | Op1282 | Op1283 | Op1284 | Op1285 | Op1286 | Op1287 | Op1288 | Op1289 | Op1290 | Op1291 | Op1292 | Op1293 | Op1294 | Op1295 | Op1296 | Op1297 | Op1298 | Op1299 | Op1300 | Op1301 | Op1302 | Op1303 | Op1304 | Op1305 | Op1306 | Op1307 | Op1308 | Op1309 | Op1310 | Op1311 | Op1312 | Op1313 | Op1314 | Op1315 | Op1316 | Op1317 | Op1318 | Op1319 | Op1320 | Op1321 | Op1322 | Op1323 | Op1324</ |
|---------|----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|----------|
|---------|----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|----------|

DBGEVALOP  
V04-000

F 4  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 B11ss-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32;1

Page 119  
(22)

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 00 | 00 | 61 | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 61 | 31 | 31 | 00 |
| 00 | 00 | 63 | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 00 | 61 | 04 | 00 | 08 |
| 08 | 00 | 00 | 00 | 6C | 04 | 31 | 00 | 08 | 00 | 00 | 00 | 6C | 31 | 04 | 00 |
| 00 | 00 | 00 | 6C | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 6C | 31 | 31 | 00 |
| 00 | 00 | 6E | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 00 | 6C | 04 | 00 | 08 |
| 08 | 00 | 00 | 00 | 90 | 31 | 04 | 00 | 08 | 00 | 00 | 00 | 8F | 31 | 08 | 00 |
| 00 | 00 | 00 | 90 | 04 | 31 | 00 | 08 | 00 | 00 | 00 | 00 | 8F | 08 | 31 | 00 |
| 00 | 00 | 8F | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 00 | 8F | 31 | 31 | 00 |
| 00 | 92 | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 90 | 04 | 04 | 00 | 08 | 00 |
| 08 | 00 | 00 | 00 | 80 | 31 | 04 | 00 | 08 | 00 | 00 | 00 | 7F | 31 | 08 | 00 |
| 00 | 00 | 00 | 80 | 04 | 31 | 00 | 08 | 00 | 00 | 00 | 00 | 7F | 08 | 31 | 00 |
| 00 | 00 | 7F | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 00 | 7F | 31 | 31 | 00 |
| 00 | 82 | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 80 | 04 | 04 | 00 | 08 | 00 |
| 08 | 00 | 00 | 00 | 88 | 31 | 04 | 00 | 08 | 00 | 00 | 00 | 87 | 31 | 08 | 00 |
| 00 | 00 | 00 | 88 | 04 | 31 | 00 | 08 | 00 | 00 | 00 | 00 | 87 | 08 | 31 | 00 |
| 00 | 00 | 87 | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 00 | 87 | 31 | 31 | 00 |
| 00 | 8A | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 88 | 04 | 04 | 00 | 08 | 00 |
| 08 | 00 | 00 | 00 | 78 | 31 | 04 | 00 | 08 | 00 | 00 | 00 | 77 | 31 | 08 | 00 |
| 00 | 00 | 00 | 78 | 04 | 31 | 00 | 08 | 00 | 00 | 00 | 00 | 77 | 08 | 31 | 00 |
| 00 | 00 | 77 | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 00 | 77 | 31 | 31 | 00 |
| 00 | 7A | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 78 | 04 | 04 | 00 | 08 | 00 |
| 04 | 00 | 00 | 00 | 97 | 04 | 04 | 00 | 08 | 00 | 00 | 00 | 97 | 08 | 08 | 00 |
| 04 | 00 | 00 | 00 | A0 | 04 | 04 | 00 | 08 | 00 | 00 | 00 | A0 | 08 | 08 | 00 |
| 04 | 00 | 00 | 00 | A3 | 04 | 04 | 00 | 08 | 00 | 00 | 00 | A3 | 08 | 08 | 00 |
| 04 | 00 | 00 | 00 | 9D | 04 | 04 | 00 | 08 | 00 | 00 | 00 | 9D | 08 | 08 | 00 |
| 08 | 00 | 00 | 00 | A6 | 31 | 04 | 00 | 08 | 00 | 00 | 00 | A6 | 31 | 31 | 00 |
| 00 | 00 | 00 | A6 | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | A6 | 04 | 31 | 00 |
| 00 | 00 | A7 | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 00 | A6 | 04 | 00 | 08 |
| 08 | 00 | 00 | 00 | AE | 31 | 04 | 00 | 08 | 00 | 00 | 00 | AE | 31 | 31 | 00 |
| 00 | 00 | 00 | AE | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | AE | 04 | 31 | 00 |
| 00 | 00 | AF | 08 | 08 | 00 | 08 | 00 | 00 | 00 | 00 | 00 | AE | 04 | 00 | 08 |

|       |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
|-------|--------|-------|--|--|--|--|--|--|--|--|--|--|--|--|--|
| 019DB |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 019EA |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 019F9 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 019FC |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A00 | P.ADG: | .LONG |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A0F |        | .BYTE |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A1E |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A2D |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A30 |        | .LONG |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A34 | P.ADH: | .BYTE |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A43 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A52 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A61 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A70 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A74 |        | .LONG |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A78 | P.ADI: | .BYTE |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A87 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01A96 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01AA5 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01AB4 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01AB8 |        | .LONG |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01ABC | P.ADJ: | .BYTE |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01ACB |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01ADA |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01AE9 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01AF8 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01AFC |        | .LONG |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B00 | P.ADK: | .BYTE |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B0F |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B1E |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B2D |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B3C |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B40 |        | .LONG |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B44 | P.ADL: | .BYTE |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B53 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B54 |        | .LONG |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B58 | P.ADM: | .BYTE |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B67 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B68 |        | .LONG |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B6C | P.ADN: | .BYTE |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B78 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B7C |        | .LONG |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B80 | P.ADO: | .BYTE |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B8F |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B90 |        | .LONG |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01B94 | P.ADP: | .BYTE |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01BA3 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01BB2 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01BC1 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01BC4 |        | .LONG |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01BC8 | P.ADQ: | .BYTE |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01BD7 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01BE6 |        |       |  |  |  |  |  |  |  |  |  |  |  |  |  |

|      |    |      |      |      |    |     |      |      |    |      |    |   |   |   |   |
|------|----|------|------|------|----|-----|------|------|----|------|----|---|---|---|---|
| 0    | 0  | 8    | 0    | 49   | 49 | 97  | 0    | 0    | 0  | 0    | 8  | 0 | 0 | 8 | - |
| 8    | 97 | 0    | 0    | 0    | 8  | 0   | 4    | 4    | 0  | 0    | 0  | 0 | 0 | 0 | - |
| 8    | 0  | 11   | 11   | 99   | 0  | 0   | 0    | 8    | 0  |      |    |   |   |   | - |
| 12   |    |      |      |      |    |     |      |      |    |      |    |   |   |   | - |
| 4    | 49 | 108  | 0    | 0    | 0  | 8   | 0    | 49   | 4  | 108  | 0  |   |   |   | - |
| 0    | 0  | 8    | 0    | 49   | 49 | 108 | 0    | 0    | 0  | 8    | 0  |   |   |   | - |
| 8    | 8  | 108  | 0    | 0    | 0  | 8   | 0    | 4    | 4  | 108  | 0  |   |   |   | - |
| 0    | 0  | 8    | 0    | 11   | 11 | 110 | 0    | 0    | 0  | 8    | 0  |   |   |   | - |
| 16   |    |      |      |      |    |     |      |      |    |      |    |   |   |   | - |
| 8    | 49 | -113 | 0    | 0    | 0  | 8   | 0    | 4    | 49 | -112 |    |   |   |   | - |
| 0    | 0  | 0    | 8    | 0    | 49 | 8   | -113 | 0    | 0  | 0    | 8  |   |   |   | - |
| 0    | 49 | 4    | -112 | 0    | 0  | 0   | 8    | 0    | 49 | 49   |    |   |   |   | - |
| -113 | 0  | 0    | 0    | 8    | 0  | 8   | 8    | -113 | 0  | 0    | 0  |   |   |   | - |
| 8    | 0  | 4    | 4    | -112 | 0  | 0   | 0    | 8    | 0  | 11   | 11 |   |   |   | - |
| -110 | 0  | 0    | 0    | 8    | 0  |     |      |      |    |      |    |   |   |   | - |
| 16   |    |      |      |      |    |     |      |      |    |      |    |   |   |   | - |
| 8    | 49 | 127  | 0    | 0    | 0  | 8   | 0    | 4    | 49 | -128 |    |   |   |   | - |
| 0    | 0  | 0    | 8    | 0    | 49 | 8   | 127  | 0    | 0  | 0    | 8  |   |   |   | - |
| 49   | 4  | -128 | 0    | 0    | 0  | 8   | 0    | 49   | 49 | 127  |    |   |   |   | - |
| 0    | 0  | 0    | 8    | 0    | 8  | 8   | 127  | 0    | 0  | 0    | 8  |   |   |   | - |
| 4    | 4  | -128 | 0    | 0    | 0  | 8   | 0    | 11   | 11 | -126 |    |   |   |   | - |
| 0    | 0  | 0    | 8    | 0    |    |     |      |      |    |      |    |   |   |   | - |
| 16   |    |      |      |      |    |     |      |      |    |      |    |   |   |   | - |
| 8    | 49 | -121 | 0    | 0    | 0  | 8   | 0    | 4    | 49 | -120 |    |   |   |   | - |
| 0    | 0  | 0    | 8    | 0    | 49 | 8   | -121 | 0    | 0  | 0    | 8  |   |   |   | - |
| 0    | 49 | 4    | -120 | 0    | 0  | 0   | 8    | 0    | 49 | 49   |    |   |   |   | - |
| -121 | 0  | 0    | 0    | 8    | 0  | 8   | 8    | -121 | 0  | 0    | 0  |   |   |   | - |
| 8    | 0  | 4    | 4    | -120 | 0  | 0   | 0    | 8    | 0  | 11   | 11 |   |   |   | - |
| -118 | 0  | 0    | 0    | 8    | 0  |     |      |      |    |      |    |   |   |   | - |
| 16   |    |      |      |      |    |     |      |      |    |      |    |   |   |   | - |
| 8    | 49 | 119  | 0    | 0    | 0  | 8   | 0    | 4    | 49 | 120  | 0  |   |   |   | - |
| 0    | 0  | 8    | 0    | 49   | 8  | 119 | 0    | 0    | 0  | 8    | 0  |   |   |   | - |
| 49   | 4  | 120  | 0    | 0    | 0  | 8   | 0    | 49   | 49 | 119  |    |   |   |   | - |
| 0    | 0  | 0    | 8    | 0    | 8  | 8   | 119  | 0    | 0  | 0    | 8  |   |   |   | - |
| 4    | 4  | 120  | 0    | 0    | 0  | 8   | 0    | 11   | 11 | 122  | 0  |   |   |   | - |
| 0    | 0  | 8    | 0    |      |    |     |      |      |    |      |    |   |   |   | - |
| 4    |    |      |      |      |    |     |      |      |    |      |    |   |   |   | - |
| 8    | 8  | -105 | 0    | 0    | 0  | 8   | 0    | 4    | 4  | -105 | 0  |   |   |   | - |
| 0    | 0  | 4    | 0    |      |    |     |      |      |    |      |    |   |   |   | - |
| 4    |    |      |      |      |    |     |      |      |    |      |    |   |   |   | - |
| 8    | 8  | -96  | 0    | 0    | 0  | 8   | 0    | 4    | 4  | -96  | 0  |   |   |   | - |
| 0    | 0  | 4    | 0    |      |    |     |      |      |    |      |    |   |   |   | - |
| 4    |    |      |      |      |    |     |      |      |    |      |    |   |   |   | - |
| 8    | 8  | -93  | 0    | 0    | 0  | 8   | 0    | 4    | 4  | -93  | 0  |   |   |   | - |
| 0    | 0  | 4    | 0    |      |    |     |      |      |    |      |    |   |   |   | - |
| 4    |    |      |      |      |    |     |      |      |    |      |    |   |   |   | - |
| 8    | 8  | -99  | 0    | 0    | 0  | 8   | 0    | 4    | 4  | -99  | 0  |   |   |   | - |
| 0    | 0  | 4    | 0    |      |    |     |      |      |    |      |    |   |   |   | - |
| 12   |    |      |      |      |    |     |      |      |    |      |    |   |   |   | - |
| 49   | 49 | -90  | 0    | 0    | 0  | 8   | 0    | 4    | 49 | -90  |    |   |   |   | - |
| 0    | 0  | 0    | 8    | 0    | 49 | 4   | -90  | 0    | 0  | 0    |    |   |   |   |   |

|  |       |                 |       |  |  |   |
|--|-------|-----------------|-------|--|--|---|
| 00   | 08    | 00              | 01BF5 |  |  | 0, 0, 8, 0, 11, 11, -81, 0, 0, 0, 8, 0      |
| 00000000                                     | 00    | 08              | 01BF8 |  |  |   |
| 08 00 00 00 AA 08 08 00 08 00 00 00 AA 31 31 | 01BFC | P.ADR:          | .LONG |  |  |   |
| 00 00 00 00 AB 08 08 00 08 00 00 00 AA 04 04 | 01C08 |                 | .BYTE |  |  | 49, 49, -86, 0, 0, 0, 8, 0, 8, 8, -86, 0, - |
|  | 01C1A |                 |       |  |  | 0, 0, 8, 0, 4, 4, -86, 0, 0, 0, 8, 0, 11, - |
| 04 00 00 00 3D 04 04 00 08 00 00 00 3D 08 08 | 01C1C | P.ADS:          | .LONG |  |  | 11, -85, 0, 0, 0, 8, 0                      |
|  | 01C2F |                 | .BYTE |  |  | 4, 8, 61, 0, 0, 0, 8, 0, 4, 4, 61, 0, 0, -  |
| 04 00 00 00 3F 04 04 00 08 00 00 00 3E 08 08 | 01C30 |                 | .LONG |  |  | 0, 4, 0                                     |
|  | 01C34 | P.ADT:          | .BYTE |  |  | 4, 8, 62, 0, 0, 0, 8, 0, 4, 4, 63, 0, 0, -  |
| 04 00 00 00 E3 04 04 00 08 00 00 00 E2 08 08 | 01C43 |                 | .LONG |  |  | 0, 4, 0                                     |
| 00 00 00 00 E5 31 31 00 08 00 00 00 E4 08 08 | 01C44 | P.ADU:          | .BYTE |  |  | 8, 8, -30, 0, 0, 0, 8, 0, 4, 4, -29, 0, -   |
|  | 01C48 |                 | .LONG |  |  | 0, 0, 4, 0, 11, 11, -28, 0, 0, 0, 11, 0, -  |
| 04 00 00 00 E7 04 04 00 08 00 00 00 E6 08 08 | 01C57 |                 | .BYTE |  |  | 49, 49, -27, 0, 0, 0, 49, 0                 |
| 00 00 00 00 E9 31 31 00 08 00 00 00 E8 08 08 | 01C66 |                 | .LONG |  |  | 8, 8, -26, 0, 0, 0, 8, 0, 4, 4, -25, 0, -   |
|  | 01C68 | P.ADV:          | .BYTE |  |  | 0, 0, 4, 0, 11, 11, -24, 0, 0, 0, 11, 0, -  |
| 04 00 00 00 EB 04 04 00 08 00 00 00 EA 08 08 | 01C7B |                 | .LONG |  |  | 49, 49, -23, 0, 0, 0, 49, 0                 |
| 00 00 00 00 ED 31 31 00 08 00 00 00 EC 08 08 | 01C8A |                 | .BYTE |  |  | 8, 8, -22, 0, 0, 0, 8, 0, 4, 4, -21, 0, -   |
|  | 01C8C | P.ADW:          | .LONG |  |  | 0, 0, 4, 0, 11, 11, -20, 0, 0, 0, 11, 0, -  |
| 04 00 00 00 EF 04 04 00 08 00 00 00 EE 08 08 | 01C90 |                 | .BYTE |  |  | 49, 49, -19, 0, 0, 0, 49, 0                 |
| 00 00 00 00 F1 31 31 00 08 00 00 00 FO 08 08 | 01C9F |                 | .LONG |  |  | 8, 8, -18, 0, 0, 0, 8, 0, 4, 4, -17, 0, -   |
|  | 01CAE | P.ADX:          | .BYTE |  |  | 0, 0, 4, 0, 11, 11, -16, 0, 0, 0, 11, 0, -  |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | 01CB0 |                 | .LONG |  |  | 49, 49, -15, 0, 0, 0, 49, 0                 |
|  | 01CB4 |                 | .BYTE |  |  | 2, 8, -38, 0, 0, 0, 49, 0                   |
| 00 31 00 00 00 00 DA 08 08                   | 01CC3 | P.ADY:          | .LONG |  |  | 2, 8, -37, 0, 0, 0, 8, 0                    |
| 00 08 00 00 00 00 DB 08 08                   | 01CD2 |                 | .LONG |  |  | 2, 8, -37, 0, 0, 0, 8, 0                    |
| 00 00 00 00 00 00 DO 31 31                   | 01CE0 | P.ADZ:          | .BYTE |  |  | 49, 49, -48, 0, 0, 0, 0, 0                  |
|  | 01CE4 | P.AEA:          | .LONG |  |  |   |
|  | 01CEC |                 | .BYTE |  |  |   |
|  | 01CF0 | C_OPINFO_TABLE: |       |  |  |   |
|  | 01CF8 |                 | .LONG |  |  |   |
| 00000000 00001830 00001CF0                   | 01D28 |                 | .BYTE |  |  | 0[48]                                       |
|  | 01D34 |                 | .LONG |  |  | 7408, 6192, 0                               |
|  | 01D35 |                 | .BYTE |  |  | 1   |
| 00000000 00001830 00001964                   | 01D38 |                 | .BYTE |  |  | 0[3]  |
|  | 01D44 |                 | .LONG |  |  | 6500, 6192, 0                               |
|  | 01D45 |                 | .BYTE |  |  | 1   |
| 00000000 00001830 00001908                   | 01D48 |                 | .BYTE |  |  | 0[3]  |
|  | 01D54 |                 | .LONG |  |  | 6408, 6192, 0                               |
|  | 01D55 |                 | .BYTE |  |  | 1   |
| 00000000 00001830 00001878                   | 01D58 |                 | .BYTE |  |  | 0[3]  |
|  | 01D64 |                 | .LONG |  |  | 6264, 6192, 0                               |
|  | 01D65 |                 | .BYTE |  |  | 1   |
| 00000000 00001830 000018A4                   | 01D68 |                 | .BYTE |  |  | 0[3]  |
|  | 01D74 |                 | .LONG |  |  | 6308, 6192, 0                               |
|  | 01D75 |                 | .BYTE |  |  | 1   |
| 00000000 00001830 000018D0                   | 01D78 |                 | .BYTE |  |  | 0[3]  |
|  | 01D84 |                 | .LONG |  |  | 6352, 6192, 0                               |
|  | 01D85 |                 | .BYTE |  |  | 1   |
| 00000000 00001830 000018EC                   | 01D88 |                 | .BYTE |  |  | 0[3]  |
|  | 01D94 |                 | .LONG |  |  | 6380, 6192, 0                               |
|  | 01D95 |                 | .BYTE |  |  | 1   |
| 00000000 00001830 000019CC                   | 01DC8 |                 | .BYTE |  |  | 0[51]                                       |
|  |       |                 | .LONG |  |  | 6604, 6192, 0                               |

|          |          |          |       |       |               |
|----------|----------|----------|-------|-------|---------------|
| 00000000 | 00001830 | 00001A00 | 01DD4 | .BYTE | 1             |
|          |          | 00#      | 01DD5 | .BYTE | 0[3]          |
|          |          | 01       | 01DD8 | .LONG | 6656, 6192, 0 |
|          |          | 00#      | 01DE4 | .BYTE | 1             |
| 00000000 | 00001830 | 00001A78 | 01DE5 | .BYTE | 0[3]          |
|          |          | 01       | 01DE8 | .LONG | 6776, 6192, 0 |
|          |          | 00#      | 01DF4 | .BYTE | 1             |
|          |          | 01       | 01DF5 | .BYTE | 0[19]         |
| 00000000 | 00001830 | 00001B00 | 01E08 | .LONG | 6912, 6192, 0 |
|          |          | 00#      | 01E14 | .BYTE | 1             |
|          |          | 01       | 01E15 | .BYTE | 0[19]         |
| 00000000 | 00001830 | 00001A34 | 01E28 | .LONG | 6708, 6192, 0 |
|          |          | 00#      | 01E34 | .BYTE | 1             |
|          |          | 01       | 01E35 | .BYTE | 0[19]         |
| 00000000 | 00001830 | 00001ABC | 01E48 | .LONG | 6844, 6192, 0 |
|          |          | 00#      | 01E54 | .BYTE | 1             |
|          |          | 01       | 01E55 | .BYTE | 0[19]         |
| 00000000 | 00001830 | 00001BFC | 01E68 | .LONG | 7164, 6192, 0 |
|          |          | 00#      | 01E74 | .BYTE | 1             |
|          |          | 01       | 01E75 | .BYTE | 0[67]         |
| 00000000 | 00001830 | 00001B94 | 01E88 | .LONG | 7060, 6192, 0 |
|          |          | 00#      | 01EC4 | .BYTE | 1             |
|          |          | 01       | 01EC5 | .BYTE | 0[3]          |
| 00000000 | 00001830 | 00001BC8 | 01EC8 | .LONG | 7112, 6192, 0 |
|          |          | 00#      | 01ED4 | .BYTE | 1             |
|          |          | 01       | 01ED5 | .BYTE | 0[3]          |
| 00000000 | 00001830 | 00001B80 | 01ED8 | .LONG | 7040, 6192, 0 |
|          |          | 00#      | 01EE4 | .BYTE | 1             |
|          |          | 01       | 01EE5 | .BYTE | 0[3]          |
| 00000000 | 00001830 | 00001B44 | 01EE8 | .LONG | 6980, 6192, 0 |
|          |          | 00#      | 01EF4 | .BYTE | 1             |
|          |          | 01       | 01EF5 | .BYTE | 0[3]          |
| 00000000 | 00001830 | 00001B58 | 01EF8 | .LONG | 7000, 6192, 0 |
|          |          | 00#      | 01F04 | .BYTE | 1             |
|          |          | 01       | 01F05 | .BYTE | 0[3]          |
| 00000000 | 00001830 | 00001B6C | 01F08 | .LONG | 7020, 6192, 0 |
|          |          | 00#      | 01F14 | .BYTE | 1             |
|          |          | 01       | 01F15 | .BYTE | 0[51]         |
| 00000000 | 00001830 | 00001988 | 01F48 | .LONG | 6584, 6192, 0 |
|          |          | 00#      | 01F54 | .BYTE | 1             |
|          |          | 01       | 01F55 | .BYTE | 0[3]          |
| 00000000 | 00001830 | 00001C20 | 01F58 | .LONG | 7200, 6192, 0 |
|          |          | 00#      | 01F64 | .BYTE | 1             |
|          |          | 01       | 01F65 | .BYTE | 0[3]          |
| 00000000 | 00001830 | 00001C34 | 01F68 | .LONG | 7220, 6192, 0 |
|          |          | 00#      | 01F74 | .BYTE | 1             |
|          |          | 01       | 01F75 | .BYTE | 0[3]          |
| 00000000 | 00001830 | 00001C78 | 01F78 | .LONG | 7384, 6192, 0 |
|          |          | 00#      | 01F84 | .BYTE | 0             |
|          |          | 01       | 01F85 | .BYTE | 0[3]          |
| 00000000 | 00001830 | 00001CE4 | 01F88 | .LONG | 7396, 6192, 0 |
|          |          | 00#      | 01F94 | .BYTE | 0             |
|          |          | 01       | 01F95 | .BYTE | 0[131]        |
| 00000000 | 00001854 | 00000000 | 02018 | .LONG | 0, 6228, 0    |
|          |          | 00#      | 02024 | .BYTE | 1             |
|          |          | 01       | 02025 | .BYTE | 0[3]          |
| 00000000 | 00001830 | 00000000 | 02028 | .LONG | 0, 6192, 0    |

| Address   | Hex                           | Dec      | Label           | Value   |
|---|-------------------------------|----------|-----------------|---|
| 00000000  | 00000000                      | 00000000 | 01 02034        | .BYTE 1   |
| 00000000  | 00000000                      | 00000000 | 00# 02035       | .BYTE 0[67]                                       |
| 00000000  | 00001830                      | 00001C48 | 01 02078        | .LONG 0, 0, 0                                     |
| 00000000  | 00001830                      | 00001C48 | 00# 02084       | .BYTE 1   |
| 00000000  | 00001830                      | 00001C6C | 01 02085        | .BYTE 0[35]                                       |
| 00000000  | 00001830                      | 00001C6C | 00# 020A8       | .LONG 7240, 6192, 0                               |
| 00000000  | 00001830                      | 00001C6C | 01 020B4        | .BYTE 1   |
| 00000000  | 00001830                      | 00001C6C | 00# 020B5       | .BYTE 0[3]  |
| 00000000  | 00001830                      | 00001C90 | 01 020B8        | .LONG 7276, 6192, 0                               |
| 00000000  | 00001830                      | 00001C90 | 00# 020C4       | .BYTE 1   |
| 00000000  | 00001830                      | 00001C90 | 01 020C5        | .BYTE 0[3]  |
| 00000000  | 00001830                      | 00001CB4 | 00# 020C8       | .LONG 7312, 6192, 0                               |
| 00000000  | 00001830                      | 00001CB4 | 01 020D4        | .BYTE 1   |
| 00000000  | 00001830                      | 00001CB4 | 00# 020D5       | .BYTE 0[3]  |
| 00000000  | 00001830                      | 00001CB4 | 01 020D8        | .LONG 7348, 6192, 0                               |
| 00000000  | 00001830                      | 00001CB4 | 00# 020E4       | .BYTE 1   |
| 00000000  | 00001830                      | 00001CB4 | 01 020E5        | .BLKB 83  |
| 00000000  | 0000213C                      | 00000002 | 02138           | .LONG 2   |
| 00000000  | 0000213C                      | 0002FF30 | 0213C P.AEB:    | .LONG 196400, 0                                   |
| 00000000  | 0000213C                      | 00000000 | 02144 COBOL_CVT | INFO_TABLE:                                       |
| 00000000  | 0000213C                      | 00000000 | 00 0214C        | .LONG 0, 8508                                     |
| 00000000  | 0000213C                      | 00000000 | 00 0214D        | .BYTE 0   |
| 00000000  | 0000213C                      | 00000000 | 00 0214E        | .BLKB 3   |
| 00000000  | 0000213C                      | 00000000 | 00 02150        | .LONG 7   |
| 1512 1511 1510 150F 1509 1505 1508 1504 1507 1503 | 0000 150B 0B0A 1513           | 00000007 | 02154 P.AEC:    | .WORD 5379, 5383, 5380, 5384, 5381, 5385, 5391, - |
| 1211 1110 100F 0F09 0905 0508 0804 0407 0703 0330 | 0000 300B 0B0A 0A15 1513 1312 | 00000008 | 02168           | 5392, 5393, 5394, 5395, 2826, 5387, 0             |
| 0A 00 00 00 07 0A 0A 00 15 00 00 00 F2 15 15      | 00 0B 00 00 00 00 08 0B 0B 00 | 00000006 | 02170           | .LONG 8   |
| 0A 00 00 00 15 0A 0A 00 15 00 00 00 F3 15 15      | 00 0B 00 00 00 00 16 0B 0B 00 | 00000006 | 02174 P.AED:    | .WORD 816, 1795, 1031, 2052, 1288, 2309, 3849, -  |
| 0A 00 00 00 31 0A 0A 00 15 00 00 00 F4 15 15      | 00 0B 00 00 00 00 32 0B 0B 00 | 00000006 | 02188           | 4111, 4368, 4625, 4882, 5395, 2581, 2826, -       |
| 0A 00 00 00 23 0A 0A 00 15 00 00 00 F5 15 15      | 00 0B 00 00 00 00 24 0B 0B 00 | 00000006 | 02194           | 12299, 0  |
| 07 00 00 00 BE 07 07 00 06 00 00 00 BD 06 06      | 00 0B 00 00 00 00 00 0B 0B 00 | 00000014 | 02198 P.AEE:    | .LONG 6   |
| 00 00 00 00 C2 1B 1B 00 08 00 00 00 BF 08 08      | 00 0B 00 00 00 00 00 0B 0B 00 | 00000014 | 021A7           | .BYTE 21, 21, -14, 0, 0, 0, 21, 0, 10, 10, 7, -   |
| 00 01 0A 09 09 00 1C 00 00 00 C3 1C 1C 00 1B      | 00 0B 00 00 00 00 00 0B 0B 00 | 00000006 | 021B0           | 0, 0, 0, 10, 0, 11, 11, 8, 0, 0, 11, 0            |
| 00 F6 15 15 00 1A 00 00 01 0C 1A 1A 00 09 00      | 00 0B 00 00 00 00 00 0B 0B 00 | 00000006 | 021B4 P.AEF:    | .LONG 6   |
| C1 0B 0B 00 0A 00 00 00 C0 0A 0A 00 15 00 00      | 00 0B 00 00 00 00 00 0B 0B 00 | 00000006 | 021C3           | .BYTE 21, 21, -13, 0, 0, 0, 21, 0, 10, 10, 21, -  |
| 07 00 00 00 B2 07 07 00 06 00 00 00 B1 06 06      | 00 0B 00 00 00 00 00 0B 0B 00 | 00000014 | 021C3           | 0, 0, 0, 10, 0, 11, 11, 22, 0, 0, 11, 0           |
| 00 00 00 00 B7 1B 1B 00 08 00 00 00 B3 08 08      | 00 0B 00 00 00 00 00 0B 0B 00 | 00000014 | 021CC           | .LONG 6   |
| 00 01 0B 09 09 00 1C 00 00 00 B8 1C 1C 00 1B      | 00 0B 00 00 00 00 00 0B 0B 00 | 00000014 | 021D0 P.AEG:    | .BYTE 21, 21, -12, 0, 0, 0, 21, 0, 10, 10, 49, -  |
| 00 F7 15 15 00 1A 00 00 01 0C 1A 1A 00 09 00      | 00 0B 00 00 00 00 00 0B 0B 00 | 00000014 | 021DF           | 0, 0, 0, 10, 0, 11, 11, 50, 0, 0, 11, 0           |
| B6 0B 0B 00 0A 00 00 00 B5 0A 0A 00 15 00 00      | 00 0B 00 00 00 00 00 0B 0B 00 | 00000014 | 021E8           | .LONG 6   |
|   |                               | 00000014 | 021EC P.AEH:    | .BYTE 21, 21, -11, 0, 0, 0, 21, 0, 10, 10, 35, -  |
|   |                               | 00000014 | 021FB           | 0, 0, 0, 10, 0, 11, 11, 36, 0, 0, 11, 0           |
|   |                               | 00000014 | 02204           | .LONG 20  |
|   |                               | 00000014 | 02208 P.AEI:    | .BYTE 6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -   |
|   |                               | 00000014 | 02217           | 0, 0, 7, 0, 8, 8, -65, 0, 0, 0, 8, 0, 27, -       |
|   |                               | 00000014 | 02226           | 27, -62, 0, 0, 0, 0, 27, 0, 28, 28, -61, 0, -     |
|   |                               | 00000014 | 02235           | 0, 0, 28, 0, 0, 0, 28, 0, 28, 28, -61, 0, -       |
|   |                               | 00000014 | 02244           | 26, 12, 1, 0, 0, 0, 26, 0, 21, 21, -10, 0, -      |
|   |                               | 00000014 | 02253           | 0, 0, 21, 0, 10, 10, -64, 0, 0, 0, 10, 0, -       |
|   |                               | 00000014 | 02258           | 11, 11, -63, 0, 0, 0, 11, 0                       |
|   |                               | 00000014 | 0225C P.AEJ:    | .LONG 20  |
|   |                               | 00000014 | 0225C           | .BYTE 6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -   |
|   |                               | 00000014 | 0226B           | 0, 0, 7, 0, 8, 8, -77, 0, 0, 0, 8, 0, 27, -       |
|   |                               | 00000014 | 0227A           | 27, -73, 0, 0, 0, 0, 27, 0, 28, 28, -72, 0, -     |
|   |                               | 00000014 | 02289           | 0, 0, 28, 0, 0, 0, 28, 0, 28, 28, -72, 0, -       |
|   |                               | 00000014 | 02298           | 26, 13, 1, 0, 0, 0, 26, 0, 21, 21, -9, 0, 0, -    |
|   |                               | 00000014 | 022A7           | 0, 21, 0, 10, 10, -75, 0, 0, 0, 10, 0, -          |

| Address     | Hex         | ASCII       | Label                         | Value  |
|-------------|-------------|-------------|-------------------------------|--|
| 28 00 00 00 | F8 15 15 00 | 28 00 00 00 | 54 0E 0E 022AC                | .LONG 8  |
| 00 00 00 63 | 0B 0B 00 28 | 00 00 00 62 | 0A 0A 00 022B0                | P.AEK: .BYTE 14, 14, 84, 0, 0, 0, 40, 0, 21, 21, -8, - |
| 28 00 00 00 | F9 15 15 00 | 28 00 00 00 | 5E 0E 0E 022BF                | .LONG 8  |
| 00 00 00 6E | 0B 0B 00 28 | 00 00 00 6D | 0A 0A 00 022CE                | P.AEL: .BYTE 14, 14, 94, 0, 0, 0, 40, 0, 21, 21, -7, - |
| 28 00 00 00 | FA 15 15 00 | 28 00 00 00 | 58 0E 0E 022D0                | .LONG 8  |
| 00 00 00 82 | 0B 0B 00 28 | 00 00 00 81 | 0A 0A 00 022D4                | P.AEM: .BYTE 14, 14, 88, 0, 0, 0, 40, 0, 21, 21, -6, - |
| 28 00 00 00 | FB 15 15 00 | 28 00 00 00 | 56 0E 0E 022E3                | .LONG 8  |
| 00 00 00 7A | 0B 0B 00 28 | 00 00 00 79 | 0A 0A 00 022F2                | P.AEN: .BYTE 14, 14, 86, 0, 0, 0, 40, 0, 21, 21, -5, - |
| 28 00 00 00 | FC 15 15 00 | 28 00 00 00 | 5C 0E 0E 02307                | .LONG 8  |
| 00 00 00 92 | 0B 0B 00 28 | 00 00 00 91 | 0A 0A 00 02316                | P.AEO: .BYTE 14, 14, 92, 0, 0, 0, 40, 0, 21, 21, -4, - |
| 28 00 00 00 | FD 15 15 00 | 28 00 00 00 | 5A 0E 0E 02318                | .LONG 8  |
| 00 00 00 8A | 0B 0B 00 28 | 00 00 00 89 | 0A 0A 00 0231C                | P.AEP: .BYTE 14, 14, 90, 0, 0, 0, 40, 0, 21, 21, -3, - |
|             |             |             | 00 28 00 00 00 AA 28 28 02373 | .LONG 2  |
|             |             |             | 00 28 00 00 00 A6 28 28 02382 | P.AEQ: .BYTE 40, 40, -86, 0, 0, 0, 40, 0               |
|             |             |             | 00 28 00 00 00 AE 28 28 02384 | .LONG 2  |
|             |             |             | 00 28 00 00 00 AE 28 28 02390 | P.AER: .BYTE 40, 40, -90, 0, 0, 0, 40, 0               |
|             |             |             | 00 28 00 00 00 AE 28 28 0239C | .LONG 2  |
|             |             |             | 00 28 00 00 00 AE 28 28 023A0 | P.AES: .BYTE 40, 40, -82, 0, 0, 0, 40, 0               |
|             |             |             | 00# 023AB                     | COBOL_OPINFO TABLE:                                    |
| 00000000    | 00002154    | 00002208    | 023E8                         | .BYTE 0[64]  |
|             |             | 01          | 023F4                         | .LONG 8712, 8532, 0                                    |
|             |             | 00#         | 023F5                         | .BYTE 1  |
| 00000000    | 00002154    | 0000225C    | 023F8                         | .BYTE 0[3]   |
|             |             | 01          | 02404                         | .LONG 8796, 8532, 0                                    |
|             |             | 00#         | 02405                         | .BYTE 1  |
| 00000000    | 00002154    | 00002198    | 02408                         | .BYTE 0[3]   |
|             |             | 01          | 02414                         | .LONG 8600, 8532, 0                                    |
|             |             | 00#         | 02415                         | .BYTE 1  |
| 00000000    | 00002154    | 000021B4    | 02418                         | .BYTE 0[3]   |
|             |             | 01          | 02424                         | .LONG 8628, 8532, 0                                    |
|             |             | 00#         | 02425                         | .BYTE 1  |
| 00000000    | 00002154    | 000021D0    | 02428                         | .BYTE 0[3]   |
|             |             | 01          | 02434                         | .LONG 8656, 8532, 0                                    |
|             |             | 00#         | 02435                         | .BYTE 1  |
| 00000000    | 00002154    | 000021EC    | 02438                         | .BYTE 0[3]   |
|             |             | 01          | 02444                         | .LONG 8684, 8532, 0                                    |
|             |             | 00#         | 02445                         | .BYTE 1  |
| 00000000    | 00002154    | 00002280    | 02478                         | .BYTE 0[51]  |
|             |             | 01          | 02484                         | .LONG 8880, 8532, 0                                    |
|             |             | 00#         | 02485                         | .BYTE 1  |
| 00000000    | 00002154    | 000022D4    | 02488                         | .BYTE 0[3]   |
|             |             | 01          | 02494                         | .LONG 8916, 8532, 0                                    |
|             |             | 00#         | 02495                         | .BYTE 1  |
|             |             |             | 02495                         | .BYTE 0[3]   |

|          |          |          |       |                   |               |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|----------|----------|----------|-------|-------------------|---------------|---------------------|------|------|------|----------|--------|-------|---|----|-------|--------|-------|---|
| 00000000 | 00002154 | 000022F8 | 02498 | .LONG             | 8952, 8532, 0 |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 01       | 024A4 | .BYTE             | 1             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 00#      | 024A5 | .BYTE             | 0[19]         |                     |      |      |      |          |        |       |   |    |       |        |       |   |
| 00000000 | 00002154 | 0000231C | 024B8 | .LONG             | 8988, 8532, 0 |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 01       | 024C4 | .BYTE             | 1             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 00#      | 024C5 | .BYTE             | 0[19]         |                     |      |      |      |          |        |       |   |    |       |        |       |   |
| 00000000 | 00002154 | 00002340 | 024D8 | .LONG             | 9024, 8532, 0 |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 01       | 024E4 | .BYTE             | 1             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 00#      | 024E5 | .BYTE             | 0[19]         |                     |      |      |      |          |        |       |   |    |       |        |       |   |
| 00000000 | 00002154 | 00002364 | 024F8 | .LONG             | 9060, 8532, 0 |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 01       | 02504 | .BYTE             | 1             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 00#      | 02505 | .BYTE             | 0[19]         |                     |      |      |      |          |        |       |   |    |       |        |       |   |
| 00000000 | 00002154 | 00002388 | 02518 | .LONG             | 9096, 8532, 0 |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 01       | 02524 | .BYTE             | 1             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 00#      | 02525 | .BYTE             | 0[3]          |                     |      |      |      |          |        |       |   |    |       |        |       |   |
| 00000000 | 00002154 | 00002394 | 02528 | .LONG             | 9108, 8532, 0 |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 01       | 02534 | .BYTE             | 1             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 00#      | 02535 | .BYTE             | 0[3]          |                     |      |      |      |          |        |       |   |    |       |        |       |   |
| 00000000 | 00002154 | 000023A0 | 02538 | .LONG             | 9120, 8532, 0 |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 01       | 02544 | .BYTE             | 1             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 00#      | 02545 | .BYTE             | 0[387]        |                     |      |      |      |          |        |       |   |    |       |        |       |   |
| 00000000 | 00002174 | 00000000 | 026C8 | .LONG             | 0, 8564, 0    |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 01       | 026D4 | .BYTE             | 1             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 00#      | 026D5 | .BYTE             | 0[3]          |                     |      |      |      |          |        |       |   |    |       |        |       |   |
| 00000000 | 00002174 | 00000000 | 026D8 | .LONG             | 0, 8564, 0    |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 01       | 026E4 | .BYTE             | 1             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 00#      | 026E5 | .BYTE             | 0[67]         |                     |      |      |      |          |        |       |   |    |       |        |       |   |
| 00000000 | 00000000 | 00000000 | 02728 | .LONG             | 0, 0, 0       |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 01       | 02734 | .BYTE             | 1             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          |          | 02735 | .BLKB             | 179           |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 00000002 | 027E8 | .LONG             | 2             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
| 0000     | 0804     | 0703     | 0602  | P.AET:            | WORD          | 1538, 1795, 2052, 0 |      |      |      |          |        |       |   |    |       |        |       |   |
|          | 00000000 | 000027EC | 027F4 | FORTTRAN_CVTINFO- | TABLE:        |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          |          |       | .LONG             | 10220, 0      |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 00       | 027FC | .BYTE             | 0             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          |          | 027FD | .BLKB             | 3             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
|          |          | 00000008 | 02800 | .LONG             | 8             |                     |      |      |      |          |        |       |   |    |       |        |       |   |
| 1D1B     | 1C1B     | 0D0B     | 1C0B  | D00A              | 1B0A          | 0B0A                | 0A0B | 0807 | 0706 | 02804    | P.AEU: | .WORD | 1798, 2055, 2568, 2826, 6922, 3082, 7179, - |    |       |        |       |   |
|          |          |          |       |                   | 0000          | 1D0C                | 0D0C | 1D1C | 0D1C | 02818    |        |       | 3339, 7195, 7451, 3356, 7452, 3340, 7436, - |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 02822    |        | .BLKB | 2   |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 02824    |        | .LONG | 4   |    |       |        |       |   |
|          |          | 0000     | 1C1B  | 1C0B              | 1B0A          | 0B0A                | 0A0B | 0807 | 0706 | 02828    | P.AEV: | .WORD | 1798, 2055, 2568, 2826, 6922, 7179, 7195, - |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 00000002 |        | .LONG | 0   |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 0000     | 0807   | 0706  | 1798, 2055, 0                               |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 02838    | P.AEW: | .WORD | 2   |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 0283C    |        | .BLKB | 2   |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 02842    |        | .LONG | 6   |    |       |        |       |   |
| 061D     | 1D0D     | 0D0C     | 0C1C  | 1C1B              | 1B0B          | 0B0A                | 0A0B | 0807 | 0706 | 02848    | P.AEX: | .WORD | 1798, 2055, 2568, 2826, 6923, 7195, 3100, - |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 0000     |        |       | 3340, 7437, 1565, 0                         |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 0285C    |        | .BLKB | 2   |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 0285E    |        | .LONG | 3   |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 00000003 |        | .WORD | 6923, 7435, 3355, 7437, 0                   |    |       |        |       |   |
|          |          | 0000     | 1D0D  | 0D1B              | 1D0B          | 1B0B                |      |      |      | 02864    | P.AEY: | .WORD | 2   |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 0286E    |        | .BLKB | 2   |    |       |        |       |   |
|          |          |          |       |                   |               |                     |      |      |      | 00000014 |        | .LONG | 20  |    |       |        |       |   |
| 07       | 00       | 00       | 00    | 03                | 07            | 07                  | 00   | 06   | 00   | 00       | 00     | 01    | 06  | 06 | 02874 | P.AEZ: | .BYTE | 6, 6, 1, 0, 0, 0, 6, 0, 7, 7, 3, 0, 0, 0, - |
| 00       | 00       | 00       | 07    | 0A                | 0A            | 00                  | 08   | 00   | 00   | 00       | 05     | 08    | 08  | 00 | 02883 |        |       | 7, 0, 8, 8, 5, 0, 0, 0, 8, 0, 10, 10, 7, -  |

DBGEVALOP  
V04-000

16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32:1

Page 125  
(22)

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|-------|
| 00 | 00 | 09 | 1B | 1B | 00 | 0B | 00 | 00 | 00 | 08 | 0B | 0B | 00 | 0A       | 02892 |
| 00 | 0B | 0C | 0C | 00 | 1C | 00 | 00 | 00 | 0A | 1C | 1C | 00 | 1B | 00       | 028A1 |
| 0D | 1D | 1D | 00 | 0D | 00 | 00 | 00 | 0C | 0D | 0D | 00 | 0C | 00 | 00       | 028B0 |
|    |    |    |    |    |    |    |    |    |    | 00 | 1D | 00 | 00 | 00       | 028BF |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    | 00000014 | 028C4 |
| 07 | 00 | 00 | 00 | 11 | 07 | 07 | 00 | 06 | 00 | 00 | 00 | 0F | 06 | 06       | 028C8 |
| 00 | 00 | 00 | 15 | 0A | 0A | 00 | 08 | 00 | 00 | 00 | 13 | 08 | 08 | 00       | 028D7 |
| 00 | 00 | 17 | 1B | 1B | 00 | 0B | 00 | 00 | 00 | 16 | 0B | 0B | 00 | 0A       | 028E6 |
| 00 | 19 | 0C | 0C | 00 | 1C | 00 | 00 | 00 | 00 | 18 | 1C | 1C | 00 | 1B       | 028F5 |
| 1B | 1D | 1D | 00 | 0D | 00 | 00 | 00 | 1A | 0D | 0D | 00 | 0C | 00 | 00       | 02904 |
|    |    |    |    |    |    |    |    |    |    | 00 | 1D | 00 | 00 | 00       | 02913 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    | 00000014 | 02918 |
| 07 | 00 | 00 | 00 | 2D | 07 | 07 | 00 | 06 | 00 | 00 | 00 | 2B | 06 | 06       | 0291C |
| 00 | 00 | 00 | 31 | 0A | 0A | 00 | 08 | 00 | 00 | 00 | 2F | 08 | 08 | 00       | 0292B |
| 00 | 00 | 33 | 1B | 1B | 00 | 0B | 00 | 00 | 00 | 32 | 0B | 0B | 00 | 0A       | 0293A |
| 00 | 35 | 0C | 0C | 00 | 1C | 00 | 00 | 00 | 00 | 34 | 1C | 1C | 00 | 1B       | 02949 |
| 37 | 1D | 1D | 00 | 0D | 00 | 00 | 00 | 36 | 0D | 0D | 00 | 0C | 00 | 00       | 02958 |
|    |    |    |    |    |    |    |    |    |    | 00 | 1D | 00 | 00 | 00       | 02967 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    | 00000014 | 0296C |
| 07 | 00 | 00 | 00 | 1F | 07 | 07 | 00 | 06 | 00 | 00 | 00 | 1D | 06 | 06       | 02970 |
| 00 | 00 | 00 | 23 | 0A | 0A | 00 | 08 | 00 | 00 | 00 | 21 | 08 | 08 | 00       | 0297F |
| 00 | 00 | 25 | 1B | 1B | 00 | 0B | 00 | 00 | 00 | 24 | 0B | 0B | 00 | 0A       | 0298E |
| 00 | 27 | 0C | 0C | 00 | 1C | 00 | 00 | 00 | 00 | 26 | 1C | 1C | 00 | 1B       | 0299D |
| 29 | 1D | 1D | 00 | 0D | 00 | 00 | 00 | 28 | 0D | 0D | 00 | 0C | 00 | 00       | 029AC |
|    |    |    |    |    |    |    |    |    |    | 00 | 1D | 00 | 00 | 00       | 029BB |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0000001A | 029C0 |
| 09 | 00 | 00 | 01 | 0A | 09 | 09 | 00 | 15 | 00 | 00 | 00 | F6 | 15 | 15       | 029C4 |
| 00 | 00 | 00 | BD | 06 | 06 | 00 | 1A | 00 | 00 | 01 | 0C | 1A | 1A | 00       | 029D3 |
| 00 | 00 | BF | 08 | 08 | 00 | 07 | 00 | 00 | 00 | BE | 07 | 07 | 00 | 06       | 029E2 |
| 00 | C1 | 0B | 0B | 00 | 0A | 00 | 00 | 00 | 00 | 0A | 0A | 00 | 08 | 00       | 029F1 |
| C3 | 1C | 1C | 00 | 1B | 00 | 00 | 00 | C2 | 1B | 1B | 00 | 0B | 00 | 00       | 02A00 |
| 0D | 00 | 00 | 0C | 00 | 00 | 00 | C4 | 0C | 0C | 00 | 1C | 00 | 00 | 00       | 02A0F |
|    | 00 | 1D | 00 | 00 | 00 | C6 | 1D | 1D | 00 | 0D | 00 | 00 | 00 | C5       | 02A1E |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0000001A | 02A2C |
| 09 | 00 | 00 | 01 | 0B | 09 | 09 | 00 | 15 | 00 | 00 | 00 | F7 | 15 | 15       | 02A30 |
| 00 | 00 | 00 | B1 | 06 | 06 | 00 | 1A | 00 | 00 | 01 | 0D | 1A | 1A | 00       | 02A3F |
| 00 | 00 | B3 | 08 | 08 | 00 | 07 | 00 | 00 | 00 | B2 | 07 | 07 | 00 | 06       | 02A4E |
| 00 | B6 | 0B | 0B | 00 | 0A | 00 | 00 | 00 | 00 | B5 | 0A | 0A | 00 | 08       | 02A5D |
| B8 | 1C | 1C | 00 | 1B | 00 | 00 | 00 | B7 | 1B | 1B | 00 | 0B | 00 | 00       | 02A6C |
| 0D | 00 | 00 | 0C | 00 | 00 | 00 | B9 | 0C | 0C | 00 | 1C | 00 | 00 | 00       | 02A7B |
|    | 00 | 1D | 00 | 00 | 00 | BB | 1D | 1D | 00 | 0D | 00 | 00 | 00 | BA       | 02A8A |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    | 00000024 | 02A98 |
| 0B | 00 | 00 | 00 | 41 | 0B | 0B | 00 | 07 | 00 | 00 | 00 | 40 | 07 | 07       | 02A9C |
| 00 | 00 | 00 | 43 | 0B | 0B | 00 | 0A | 00 | 00 | 00 | 42 | 0A | 0B | 00       | 02AAB |
| 00 | 00 | 45 | 1C | 0B | 00 | 1B | 00 | 00 | 00 | 44 | 1B | 0B | 00 | 0B       | 02ABA |
| 00 | 47 | 0D | 0B | 00 | 0C | 00 | 00 | 00 | 00 | 46 | 0C | 0B | 00 | 1C       | 02AC9 |
| 49 | 0A | 0A | 00 | 1D | 00 | 00 | 00 | 48 | 1D | 0B | 00 | 0D | 00 | 00       | 02AD8 |
| 0A | 0B | 00 | 0B | 00 | 00 | 00 | 4A | 0B | 0A | 00 | 0A | 00 | 00 | 00       | 02AE7 |
| 1B | 00 | 0B | 00 | 00 | 00 | 4C | 0B | 0B | 00 | 0B | 00 | 00 | 00 | 4B       | 02AF6 |

P.AFA: .LONG  
.BYTE

P.AFB: .LONG  
.BYTE

P.AFC: .LONG  
.BYTE

P.AFD: .LONG  
.BYTE

P.AFE: .LONG  
.BYTE

P.AFF: .LONG  
.BYTE

|    |     |     |     |    |    |     |     |    |     |     |    |    |   |  |  |
|----|-----|-----|-----|----|----|-----|-----|----|-----|-----|----|----|---|--|--|
| 0  | 0   | 0   | 10  | 0  | 11 | 11  | 8   | 0  | 0   | 0   | 11 | -  |   |  |  |
| 0  | 27  | 0   | 27  | 0  | 0  | 0   | 27  | 0  | 28  | 28  | 10 | -  |   |  |  |
| 0  | 0   | 0   | 28  | 0  | 12 | 12  | 11  | 0  | 0   | 0   | 12 | -  |   |  |  |
| 0  | 13  | 13  | 12  | 0  | 0  | 0   | 13  | 0  | 29  | 29  | -  |    |   |  |  |
| 13 | 0   | 0   | 0   | 29 | 0  |     |     |    |     |     |    |    |   |  |  |
| 20 |     |     |     |    |    |     |     |    |     |     |    |    |   |  |  |
| 6  | 6   | 15  | 0   | 0  | 0  | 6   | 0   | 7  | 7   | 17  | 0  | 0  | - |  |  |
| 0  | 7   | 0   | 8   | 8  | 19 | 0   | 0   | 0  | 8   | 0   | 10 | 10 | - |  |  |
| 21 | 0   | 0   | 0   | 10 | 0  | 11  | 11  | 22 | 0   | 0   | 0  | -  |   |  |  |
| 11 | 0   | 27  | 27  | 23 | 0  | 0   | 0   | 27 | 0   | 28  | 0  | -  |   |  |  |
| 28 | 24  | 0   | 0   | 0  | 28 | 0   | 12  | 12 | 25  | 0   | 0  | -  |   |  |  |
| 0  | 12  | 0   | 13  | 13 | 26 | 0   | 0   | 0  | 13  | 0   | 29 | -  |   |  |  |
| 26 | 27  | 0   | 0   | 0  | 29 | 0   |     |    |     |     |    |    |   |  |  |
| 20 |     |     |     |    |    |     |     |    |     |     |    |    |   |  |  |
| 6  | 6   | 43  | 0   | 0  | 0  | 6   | 0   | 7  | 7   | 45  | 0  | 0  | - |  |  |
| 0  | 7   | 0   | 8   | 8  | 47 | 0   | 0   | 0  | 8   | 0   | 10 | 10 | - |  |  |
| 49 | 0   | 0   | 0   | 10 | 0  | 11  | 11  | 50 | 0   | 0   | 0  | -  |   |  |  |
| 11 | 0   | 27  | 27  | 51 | 0  | 0   | 0   | 27 | 0   | 28  | 0  | -  |   |  |  |
| 28 | 52  | 0   | 0   | 0  | 28 | 0   | 12  | 12 | 53  | 0   | 0  | -  |   |  |  |
| 0  | 12  | 0   | 13  | 13 | 54 | 0   | 0   | 0  | 13  | 0   | 29 | -  |   |  |  |
| 26 | 55  | 0   | 0   | 0  | 29 | 0   |     |    |     |     |    |    |   |  |  |
| 20 |     |     |     |    |    |     |     |    |     |     |    |    |   |  |  |
| 6  | 6   | 29  | 0   | 0  | 0  | 6   | 0   | 7  | 7   | 31  | 0  | 0  | - |  |  |
| 0  | 7   | 0   | 8   | 8  | 33 | 0   | 0   | 0  | 8   | 0   | 10 | 10 | - |  |  |
| 35 | 0   | 0   | 0   | 10 | 0  | 11  | 11  | 36 | 0   | 0   | 0  | -  |   |  |  |
| 11 | 0   | 27  | 27  | 37 | 0  | 0   | 0   | 27 | 0   | 28  | 0  | -  |   |  |  |
| 28 | 38  | 0   | 0   | 0  | 28 | 0   | 12  | 12 | 39  | 0   | 0  | -  |   |  |  |
| 0  | 12  | 0   | 13  | 13 | 40 | 0   | 0   | 0  | 13  | 0   | 29 | -  |   |  |  |
| 26 | 41  | 0   | 0   | 0  | 29 | 0   |     |    |     |     |    |    |   |  |  |
| 26 |     |     |     |    |    |     |     |    |     |     |    |    |   |  |  |
| 21 | 21  | -10 | 0   | 0  | 0  | 21  | 0   | 9  | 9   | 10  | 1  | -  |   |  |  |
| 0  | 0   | 0   | 0   | 26 | 26 | 12  | 1   | 0  | 0   | 26  | 0  | -  |   |  |  |
| 6  | 6   | -67 | 0   | 0  | 0  | 6   | 0   | 7  | 7   | -66 | 0  | -  |   |  |  |
| 0  | 0   | 7   | 0   | 8  | 8  | -65 | 0   | 0  | 0   | 8   | 0  | 10 | - |  |  |
| 10 | -64 | 0   | 0   | 0  | 10 | 0   | 11  | 11 | -63 | 0   | 0  | -  |   |  |  |
| 0  | 0   | 11  | 0   | 27 | 27 | -62 | 0   | 0  | 0   | 27  | 0  | -  |   |  |  |
| 28 | 28  | -61 | 0   | 0  | 0  | 28  | 0   | 12 | 12  | -60 | 0  | -  |   |  |  |
| 0  | 0   | 0   | 12  | 0  | 13 | 13  | -59 | 0  | 0   | 0   | 13 | -  |   |  |  |
| 0  | 29  | 29  | -58 | 0  | 0  | 0   | 29  | 0  |     |     |    |    |   |  |  |
| 26 |     |     |     |    |    |     |     |    |     |     |    |    |   |  |  |
| 21 | 21  | -9  | 0   | 0  | 0  | 21  | 0   | 9  | 9   | 11  | 1  | -  |   |  |  |
| 0  | 0   | 0   | 0   | 26 | 26 | 13  | 1   | 0  | 0   | 26  | 0  | -  |   |  |  |
| 6  | 6   | -79 | 0   | 0  | 0  | 6   | 0   | 7  | 7   | -78 | 0  | -  |   |  |  |
| 0  | 0   | 7   | 0   | 8  | 8  | -77 | 0   | 0  | 0   | 8   | 0  | 10 | - |  |  |
| 10 | -75 | 0   | 0   | 0  | 10 | 0   | 11  | 11 | -74 | 0   | 0  | -  |   |  |  |
| 0  | 0   | 11  | 0   | 27 | 27 | -73 | 0   | 0  | 0   | 27  | 0  | -  |   |  |  |
| 28 | 28  | -72 | 0   | 0  | 0  | 28  | 0   | 12 | 12  | -71 | 0  | -  |   |  |  |
| 0  | 0   | 0   | 12  | 0  | 13 | 13  | -70 | 0  | 0   | 0   | 13 | -  |   |  |  |
| 0  | 29  | 29  | -69 | 0  | 0  | 0   | 29  | 0  |     |     |    |    |   |  |  |
| 36 |     |     |     |    |    |     |     |    |     |     |    |    |   |  |  |
| 7  | 7   | 64  | 0   | 0  | 0  | 7   | 0   | 8  | 8   | 65  | 0  | 0  | - |  |  |
| 0  | 8   | 0   | 8   | 10 | 66 | 0   | 0   | 0  | 0   | 10  | 0  | 8  | - |  |  |
| 11 | 67  | 0   | 0   | 0  | 11 | 0   | 8   | 27 | 68  | 0   | 0  | -  |   |  |  |
| 0  | 27  | 0   | 8   | 28 | 69 | 0   | 0   | 0  | 28  | 0   | 8  | -  |   |  |  |
| 12 | 70  | 0   | 0   | 0  | 12 | 0   | 8   | 13 | 71  | 0   | 0  | -  |   |  |  |
| 0  | 13  | 0   | 8   | 29 | 72 | 0   | 0   | 0  | 29  | 0   | 10 | -  |   |  |  |
| 10 | 73  | 0   | 0   | 0  | 10 | 0   | 10  | 11 | 74  | 0   | 0  | -  |   |  |  |

Page 126  
(22)

N 4  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

Page 127  
(22)

| 07 | 00 | 00 | 00 | 96 | 07 | 07 | 00 | 06 | 00 | 00 | 00 | 95 | 06 | 06 | 02D20 | P.AFD: | .BYTE | 6, 6, -107, 0, 0, 0, 6, 0, 7, 7, -106, 0, - |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|--------|-------|---|
| 07 | 00 | 00 | 00 | 9F | 07 | 07 | 00 | 06 | 00 | 00 | 00 | 9E | 06 | 06 | 02D2F |        |       | 0, 0, 7, 0, 8, 8, -105, 0, 0, 0, 8, 0       |
| 07 | 00 | 00 | 00 | A2 | 07 | 07 | 00 | 06 | 00 | 00 | 00 | A0 | 08 | 08 | 02D38 | P.AFP: | .LONG | 6, 6, -98, 0, 0, 0, 6, 0, 7, 7, -97, 0, -   |
| 07 | 00 | 00 | 00 | 99 | 07 | 07 | 00 | 06 | 00 | 00 | 00 | A1 | 06 | 06 | 02D3C |        |       | 0, 0, 7, 0, 8, 8, -96, 0, 0, 0, 8, 0        |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | A3 | 08 | 08 | 02D48 | P.AFQ: | .BYTE | 6, 6, -95, 0, 0, 0, 6, 0, 7, 7, -94, 0, -   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 98 | 06 | 06 | 02D54 |        |       | 0, 0, 7, 0, 8, 8, -93, 0, 0, 0, 8, 0        |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D58 | P.AFR: | .LONG | 6, 6, -104, 0, 0, 0, 6, 0, 7, 7, -103, 0, - |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D67 |        |       | 0, 0, 7, 0, 8, 8, -102, 0, 0, 0, 8, 0       |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D70 |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D74 |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D77 |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D7A |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D7D |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D7F |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D82 |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D85 |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D88 |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D8B |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D8E |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D91 |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D94 |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D97 |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D9A |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D9D |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02D9F |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02DA2 |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02DA5 |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02DA8 |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02DAB |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02DAE |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02DAD |        |       |   |
|    |    |    |    |    |    | 07 | 00 | 06 | 00 | 00 | 00 | 9A | 08 | 08 | 02DAF |        |       |   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        |       |   |

|  |           |          |                      |        |   |  |  |
|--|-----------|----------|----------------------|--------|---|--|--|
| 00000000   | 0000283C  | 00002D3C | 00#                  | 02F19  | .BYTE                                     | 0[3]   |  |
|  |           | 01       | 02F1C                | .LONG  | 11580, 10300, 0                           |  |  |
|  |           | 00#      | 02F28                | .BYTE  | 1   |  |  |
| 00000000   | 0000283C  | 00002D58 | 00#                  | 02F29  | .BYTE                                     | 0[3]   |  |
|  |           | 01       | 02F2C                | .LONG  | 11608, 10300, 0                           |  |  |
|  |           | 00#      | 02F38                | .BYTE  | 1   |  |  |
| 00000000   | 0000283C  | 00002D74 | 00#                  | 02F39  | .BYTE                                     | 0[3]   |  |
|  |           | 01       | 02F3C                | .LONG  | 11636, 10300, 0                           |  |  |
|  |           | 00#      | 02F48                | .BYTE  | 1   |  |  |
| 00000000   | 00000000  | 00002B30 | 00#                  | 02F49  | .BYTE                                     | 0[115]   |  |
|  |           | 01       | 02FBC                | .LONG  | 11056, 0, 0                               |  |  |
|  |           | 00#      | 02FC8                | .BYTE  | 1   |  |  |
| 00002864   | 00002848  | 00000000 | 00#                  | 02FC9  | .BYTE                                     | 0[227]   |  |
|  |           | 01       | 030AC                | .LONG  | 0, 10312, 10340                           |  |  |
|  |           | 00#      | 030B8                | .BYTE  | 1   |  |  |
| 00000000   | 0000283C  | 00000000 | 00#                  | 030B9  | .BYTE                                     | 0[3]   |  |
|  |           | 01       | 030BC                | .LONG  | 0, 10300, 0                               |  |  |
|  |           | 00#      | 030C8                | .BYTE  | 1   |  |  |
| 00000000   | 00000000  | 00000000 | 00#                  | 030C9  | .BYTE                                     | 0[67]  |  |
|  |           | 01       | 0310C                | .LONG  | 0, 0, 0                                   |  |  |
|  |           |          | 03118                | .BYTE  | 1   |  |  |
|  |           |          | 03119                | .BLKB  | 179                                       |  |  |
|  |           | 00000001 | 031CC                | .LONG  | 1   |  |  |
|  | 0000 0816 | 031D0    | P.AFS:               | .WORD  | 2070, 0                                   |  |  |
| 00000000   | 000031D0  | 031D4    | MACRO_CVTINFO_TABLE: | .LONG  | 12752, 0                                  |  |  |
|  |           | 00       | 031DC                | .BYTE  | 0   |  |  |
|  |           |          | 031DD                | .BLKB  | 3   |  |  |
|  |           | 00000003 | 031E0                | .LONG  | 3   |  |  |
| 0000 0807 0706 0804 0403 0302                      |           |          | 031E4                | P.AFT: | .WORD                                     | 770, 1027, 2052, 1798, 2055, 0                 |  |
| 00000007   |           |          | 031F0                | .LONG  | 7   |  |  |
| 1601 0129 292A 2A22 2204 0408 0803 0307 0702 0206  |           |          | 031F4                | P.AFU: | .WORD                                     | 518, 1794, 775, 2051, 1032, 8708, 10786, -     |  |
| 0000 0617 1716                                     |           |          | 03208                |        |   | 10538, 297, 5633, 5910, 1559, 0                |  |
|  |           |          | 0320E                | .BLKB  | 2   |  |  |
|  |           | 00000002 | 03210                | .LONG  | 2   |  |  |
| 00 08 00 00 00 05 08 08                            |           |          | 03214                | P.AFV: | .BYTE                                     | 8, 8, 5, 0, 0, 0, 8, 0                         |  |
|  |           | 00000002 | 0321C                | .LONG  | 2   |  |  |
| 00 08 00 00 00 13 08 08                            |           |          | 03220                | P.AFW: | .BYTE                                     | 8, 8, 19, 0, 0, 0, 8, 0                        |  |
|  |           | 00000002 | 03228                | .LONG  | 2   |  |  |
| 00 08 00 00 00 2F 08 08                            |           |          | 0322C                | P.AFX: | .BYTE                                     | 8, 8, 47, 0, 0, 0, 8, 0                        |  |
|  |           | 00000002 | 03234                | .LONG  | 2   |  |  |
| 00 08 00 00 00 21 08 08                            |           |          | 03238                | P.AFY: | .BYTE                                     | 8, 8, 33, 0, 0, 0, 8, 0                        |  |
|  |           | 00000002 | 03240                | .LONG  | 2   |  |  |
| 00 08 00 00 00 3D 08 08                            |           |          | 03244                | P.AFZ: | .BYTE                                     | 8, 8, 61, 0, 0, 0, 8, 0                        |  |
|  |           | 00000002 | 0324C                | .LONG  | 2   |  |  |
| 00 08 00 00 00 39 08 08                            |           |          | 03250                | P.AGA: | .BYTE                                     | 8, 8, 57, 0, 0, 0, 8, 0                        |  |
|  |           | 00000014 | 03258                | .LONG  | 20  |  |  |
| 07 00 00 00 BE 07 07 00 06 00 00 00 BD 06 06       |           |          | 0325C                | P.AGB: | .BYTE                                     | 6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -      |  |
| 00 00 00 00 C1 0B 0B 00 0A 00 00 00 C0 0A 0A 00    |           |          | 03268                |        |   | 0, 0, 7, 0, 10, 10, -64, 0, 0, 0, 10, 0, -     |  |
| 00 00 00 00 C3 1C 1C 00 1B 00 00 00 C2 1B 1B 00 0B |           |          | 0327A                |        |   | 11, 11, -63, 0, 0, 0, 11, 0, 27, 27, -62, 0, - |  |
| 01 0A 09 09 00 15 00 00 00 F6 15 15 00 1C 00       |           |          | 03289                |        |   | 0, 0, 0, 27, 0, 28, 28, -61, 0, 0, 0, 28, 0, - |  |
| BF 08 08 00 1A 00 00 01 0C 1A 1A 00 09 00 00       |           |          | 03298                |        |   | 0, 21, 21, -10, 0, 0, 0, 21, 0, 9, 9, 10, -    |  |
|  |           |          | 032A7                |        |   | 1, 0, 0, 9, 0, 26, 26, 12, 1, 0, 0, 26, -      |  |
|  |           |          |                      |        |   | 0, 8, 8, -65, 0, 0, 0, 8, 0                    |  |
|  |           | 00000014 | 032AC                | P.AGC: | .LONG                                     | 20   |  |
| 07 00 00 00 B2 07 07 00 06 00 00 00 B1 06 06       |           |          | 032B0                | .BYTE  | 6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, - |  |  |
| 00 00 00 00 B6 0B 0B 00 0A 00 00 00 B5 0A 0A 00    |           |          | 032BF                |        |   | 0, 0, 7, 0, 10, 10, -75, 0, 0, 0, 10, 0, -     |  |

```

00 00 B8 1C 1C 00 1B 00 00 00 B7 1B 1B 00 0B 032CE
01 08 09 09 00 15 00 00 00 F7 15 15 00 1C 00 032DD
B3 08 08 00 1A 00 00 01 0D 1A 1A 00 09 00 00 032EC
00 08 00 00 00 00 00 00 00 00 08 00 00 00 032FB
00 08 00 00 00 61 08 08 00000002 03300
00 08 00 00 00 6C 08 08 00000002 03304 P.AGD: .LONG
00 08 00 00 00 8F 08 08 00000002 0330C .LONG
00 08 00 00 00 7F 08 08 00000002 03310 P.AGE: .BYTE
00 08 00 00 00 87 08 08 00000002 03318 .LONG
00 08 00 00 00 77 08 08 00000002 0331C P.AGF: .BYTE
00 08 00 00 00 90 08 08 00000002 03324 .LONG
00 08 00 00 00 80 08 08 00000002 03328 P.AGG: .BYTE
00 08 00 00 00 88 08 08 00000002 03330 .LONG
00 08 00 00 00 78 08 08 00000002 03334 P.AGH: .BYTE
00 08 00 00 00 97 08 08 00000002 0333C .LONG
00 08 00 00 00 A0 08 08 00000002 03340 P.AGI: .BYTE
00 08 00 00 00 A3 08 08 00000002 03348 .LONG
00 08 00 00 00 9A 08 08 00000002 0334C P.AGJ: .BYTE
00 08 00 00 00 9D 08 08 00000002 03354 .LONG
00 08 00 00 00 D1 08 08 00000002 03358 P.AGK: .BYTE
00 08 00 00 00 02 06 06 00000014 03360 .LONG
00 08 00 00 00 04 04 00 00 00 00 00 00 00 00 03364 P.AGL: .BYTE
00 08 00 00 00 22 22 00 08 00 00 00 00 00 00 0336C .LONG
00 08 00 00 00 2A 2A 00 08 00 00 00 00 00 00 03370 P.AGM: .BYTE
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 03378 .LONG
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 0337C P.AGN: .BYTE
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 03384 .LONG
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 03388 P.AGO: .BYTE
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 03390 .LONG
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 03394 P.AGP: .BYTE
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 0339C .LONG
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 033A0 P.AGQ: .BYTE
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 033A8 .LONG
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 033AC P.AGR: .BYTE
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 033B4 .LONG
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 033B8 P.AGS: .BYTE
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 033C0 .LONG
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 033C4 P.AGT: .BYTE
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 033D3
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 033E2
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 033F1
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 03400
00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 0340F

```

```

08 00 00 00 CF 07 07 00 08 00 00 00 CF 06 06 033C4
00 00 00 CF 03 03 00 08 00 00 00 CF 02 02 00 00 033D3
00 00 CF 01 01 00 08 00 00 00 CF 04 04 00 08 033E2
00 CF 29 29 00 08 00 00 00 CF 22 22 00 08 00 00 033F1
CF 08 08 00 08 00 00 00 CF 2A 2A 00 08 00 00 03400
00 08 00 00 00 00 00 00 00 00 00 00 00 00 0340F

```

00# 03414 MACRO\_OPINFO TABLE:

```

00000000 000031E4 000033C4 03444 .BYTE 0[48]
00000000 000031E4 0000325C 03450 .LONG 13252, 12772, 0
00000000 000031E4 0000325C 03451 .BYTE 1
00000000 000031E4 0000325C 03454 .LONG 0[3]
00000000 000031E4 00003280 03460 .LONG 12892, 12772, 0
00000000 000031E4 00003280 03461 .BYTE 0
00000000 000031E4 00003280 03464 .LONG 0[3]
00000000 000031E4 00003280 03466 .LONG 12976, 12772, 0
00000000 000031E4 00003280 03467 .BYTE 0
00000000 000031E4 00003280 03468 .LONG 0[3]
00000000 000031E4 00003280 03469 .LONG 12820, 12772, 0

```

```

11, 11, -74, 0, 0, 0, 11, 0, 27, 27, -73, -
0, 0, 0, 27, 0, 28, 28, -72, 0, 0, 0, 28, -
0, 21, 21, -9, 0, 0, 0, 21, 0, 9, 9, 11, -
1, 0, 0, 0, 0, 26, 26, 13, 1, 0, 0, 26, -
0, 8, 8, -77, 0, 0, 0, 8, 0
8, 97, 0, 0, 0, 8, 0
8, 108, 0, 0, 0, 8, 0
8, -113, 0, 0, 0, 8, 0
8, 127, 0, 0, 0, 8, 0
8, -121, 0, 0, 0, 8, 0
8, 119, 0, 0, 0, 8, 0
8, -112, 0, 0, 0, 8, 0
8, -128, 0, 0, 0, 8, 0
8, -120, 0, 0, 0, 8, 0
8, 120, 0, 0, 0, 8, 0
8, -105, 0, 0, 0, 8, 0
8, -96, 0, 0, 0, 8, 0
8, -93, 0, 0, 0, 8, 0
8, -102, 0, 0, 0, 8, 0
8, -99, 0, 0, 0, 8, 0
8, -47, 0, 0, 0, 8, 0
6, 6, -49, 0, 0, 0, 8, 0, 7, 7, -49, 0, -
0, 0, 8, 0, 2, 2, -49, 0, 0, 0, 8, 0, 3, -
0, -49, 0, 0, 0, 8, 0, 4, 4, -49, 0, 0, -
0, 8, 0, 1, 1, -49, 0, 0, 0, 8, 0, 34, -
34, -49, 0, 0, 0, 8, 0, 41, 41, -49, 0, -
0, 0, 8, 0, 42, 42, -49, 0, 0, 0, 8, 0, -
8, 8, -49, 0, 0, 0, 8, 0

```

|          |          |          |       |       |                 |
|----------|----------|----------|-------|-------|-----------------|
|          |          | 00       | 03480 | .BYTE | 0               |
|          |          | 00#      | 03481 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 00003220 | 03484 | .LONG | 12832, 12772, 0 |
|          |          | 00       | 03490 | .BYTE | 0               |
|          |          | 00#      | 03491 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 0000322C | 03494 | .LONG | 12844, 12772, 0 |
|          |          | 00       | 034A0 | .BYTE | 0               |
|          |          | 00#      | 034A1 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 00003238 | 034A4 | .LONG | 12856, 12772, 0 |
|          |          | 00       | 034B0 | .BYTE | 0               |
|          |          | 00#      | 034B1 | .BYTE | 0[51]           |
| 00000000 | 000031E4 | 00003304 | 034E4 | .LONG | 13060, 12772, 0 |
|          |          | 00       | 034F0 | .BYTE | 0               |
|          |          | 00#      | 034F1 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 00003310 | 034F4 | .LONG | 13072, 12772, 0 |
|          |          | 00       | 03500 | .BYTE | 0               |
|          |          | 00#      | 03501 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 00003328 | 03504 | .LONG | 13096, 12772, 0 |
|          |          | 00       | 03510 | .BYTE | 0               |
|          |          | 00#      | 03511 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 00003358 | 03514 | .LONG | 13144, 12772, 0 |
|          |          | 00       | 03520 | .BYTE | 0               |
|          |          | 00#      | 03521 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 00003340 | 03524 | .LONG | 13120, 12772, 0 |
|          |          | 00       | 03530 | .BYTE | 0               |
|          |          | 00#      | 03531 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 00003370 | 03534 | .LONG | 13168, 12772, 0 |
|          |          | 00       | 03540 | .BYTE | 0               |
|          |          | 00#      | 03541 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 0000331C | 03544 | .LONG | 13084, 12772, 0 |
|          |          | 00       | 03550 | .BYTE | 0               |
|          |          | 00#      | 03551 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 0000334C | 03554 | .LONG | 13132, 12772, 0 |
|          |          | 00       | 03560 | .BYTE | 0               |
|          |          | 00#      | 03561 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 00003334 | 03564 | .LONG | 13108, 12772, 0 |
|          |          | 00       | 03570 | .BYTE | 0               |
|          |          | 00#      | 03571 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 00003364 | 03574 | .LONG | 13156, 12772, 0 |
|          |          | 00       | 03580 | .BYTE | 0               |
|          |          | 00#      | 03581 | .BYTE | 0[115]          |
| 00000000 | 000031E4 | 000033AC | 035F4 | .LONG | 13228, 12772, 0 |
|          |          | 00       | 03600 | .BYTE | 0               |
|          |          | 00#      | 03601 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 0000337C | 03604 | .LONG | 13180, 12772, 0 |
|          |          | 00       | 03610 | .BYTE | 0               |
|          |          | 00#      | 03611 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 00003388 | 03614 | .LONG | 13192, 12772, 0 |
|          |          | 00       | 03620 | .BYTE | 0               |
|          |          | 00#      | 03621 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 00003394 | 03624 | .LONG | 13204, 12772, 0 |
|          |          | 00       | 03630 | .BYTE | 0               |
|          |          | 00#      | 03631 | .BYTE | 0[3]            |
| 00000000 | 000031E4 | 000033A0 | 03634 | .LONG | 13216, 12772, 0 |
|          |          | 00       | 03640 | .BYTE | 0               |
|          |          | 00#      | 03641 | .BYTE | 0[35]           |
| 00000000 | 000031E4 | 00003250 | 03664 | .LONG | 12880, 12772, 0 |

.....



| DBGEVALOP<br>V04-000                                  | 00 1C 00 | 039A5 |              | 0. 27. 0. 28. 28. 78. 0. 0. 0. 28. 0        |
|---|----------|-------|--------------|---|
| 08 00 00 00 24 08 08 00 0A 00 00 00 23 0A 0A 039A8    | 00000008 | 039A8 | P.AHD: .LONG | 8   |
| 00 00 00 26 1C 1C 00 1B 00 00 00 25 1B 1B 00 039AC    | 00000008 | 039AC | .BYTE        | 10. 10. 35. 0. 0. 0. 10. 0. 11. 11. 36. -   |
| 07 00 00 00 BE 07 07 00 06 00 00 00 BD 06 06 039BB    | 00000016 | 039BB |              | 0. 0. 0. 11. 0. 27. 27. 37. 0. 0. 0. 27. -  |
| 00 00 01 0A 09 09 00 15 00 00 00 F6 15 15 00 039CA    | 00000016 | 039CA | P.AHE: .LONG | 0. 28. 28. 38. 0. 0. 0. 28. 0               |
| 00 00 BF 08 08 00 1A 00 00 01 0C 1A 1A 00 09 039CC    | 00000016 | 039CC | .BYTE        | 22  |
| 00 00 0A 0A 00 04 00 00 00 BF 04 04 00 08 00 039D0    | 00000016 | 039D0 |              | 6. 6. -67. 0. 0. 0. 6. 0. 7. 7. -66. 0. -   |
| C2 1B 1B 00 08 00 00 00 C1 0B 0B 00 0A 00 00 039DF    | 00000016 | 039DF |              | 0. 0. 7. 0. 21. 21. -10. 0. 0. 0. 21. 0. -  |
| 00 1C 00 00 00 00 00 C3 1C 1C 00 1B 00 00 00 039EE    | 00000016 | 039EE |              | 9. 9. 10. 1. 0. 0. 9. 0. 26. 26. 12. 1. -   |
|   | 00000016 | 039FD |              | 0. 0. 26. 0. 8. 8. -65. 0. 0. 0. 8. 0. 4. - |
|   | 00000016 | 03A0C |              | 4. -65. 0. 0. 0. 4. 0. 10. 10. -64. 0. 0. - |
|   | 00000016 | 03A1B |              | 0. 10. 0. 11. 11. -63. 0. 0. 0. 11. 0. -    |
|   | 00000016 | 03A28 | P.AHF: .LONG | 27. 27. -62. 0. 0. 0. 27. 0. 28. 28. -61. - |
|   | 00000016 | 03A2C | .BYTE        | 0. 0. 28. 0                                 |
| 07 00 00 00 B2 07 07 00 06 00 00 00 B1 06 06 03A3B    | 00000016 | 03A3B |              | 22  |
| 00 00 01 0B 09 09 00 15 00 00 00 F7 15 15 00 03A4A    | 00000016 | 03A4A |              | 6. 6. -79. 0. 0. 0. 6. 0. 7. 7. -78. 0. -   |
| 00 00 B3 08 08 00 1A 00 00 01 0D 1A 1A 00 09 03A59    | 00000016 | 03A59 |              | 0. 0. 7. 0. 21. 21. -4. 0. 0. 0. 21. 0. -   |
| 00 B5 0A 0A 00 04 00 00 00 B3 04 04 00 08 00 03A68    | 00000016 | 03A68 |              | 9. 9. 11. 1. 0. 0. 9. 0. 26. 26. 13. 1. -   |
| B7 1B 1B 00 08 00 00 00 B6 0B 0B 00 0A 00 00 03A77    | 00000016 | 03A77 |              | 0. 0. 26. 0. 8. 8. -77. 0. 0. 0. 8. 0. 4. - |
|   | 00000016 | 03A84 | P.AHG: .LONG | 4. -77. 0. 0. 0. 4. 0. 10. 10. -75. 0. 0. - |
|   | 00000016 | 03A88 | .BYTE        | 0. 10. 0. 11. 11. -74. 0. 0. 0. 11. 0. -    |
|   | 00000016 | 03A97 |              | 27. 27. -73. 0. 0. 0. 27. 0. 28. 28. -72. - |
|   | 00000016 | 03A98 |              | 0. 0. 28. 0                                 |
| 04 00 00 00 22 04 04 00 08 00 00 00 21 08 08 03A9C    | 00000016 | 03A9C | P.AHH: .LONG | 4   |
|   | 00000016 | 03A9B | .BYTE        | 8. 8. 33. 0. 0. 0. 8. 0. 4. 4. 34. 0. 0. -  |
| 04 00 00 00 3A 04 04 00 08 00 00 00 39 08 08 03AAB    | 00000016 | 03AAB |              | 0. 4. 0                                     |
|   | 00000016 | 03AAC | P.AHI: .LONG | 4   |
| 04 00 00 00 3C 04 04 00 08 00 00 00 3B 08 08 03AB0    | 00000016 | 03AB0 | .BYTE        | 8. 8. 57. 0. 0. 0. 8. 0. 4. 4. 58. 0. 0. -  |
|   | 00000016 | 03ABF |              | 0. 4. 0                                     |
|   | 00000016 | 03AC0 | P.AHJ: .LONG | 4   |
| 28 00 00 00 61 28 28 00 28 00 02 00 D3 33 33 03AC4    | 00000016 | 03AC4 | .BYTE        | 8. 8. 59. 0. 0. 0. 8. 0. 4. 4. 60. 0. 0. -  |
| 00 00 00 53 25 25 00 28 00 01 00 61 2F 2F 00 03AD3    | 00000016 | 03AD3 |              | 0. 4. 0                                     |
| 00 00 61 04 04 00 28 00 00 00 61 08 08 00 28 00 03AE2 | 00000016 | 03AE2 |              | 26  |
| 00 63 0B 0B 00 28 00 00 00 62 0A 0A 00 28 00 00 03AF1 | 00000016 | 03AF1 |              | 51. 51. -45. 0. 2. 0. 40. 0. 40. 40. 97. -  |
| 65 1C 1C 00 28 00 00 00 64 1B 1B 00 28 00 00 03B00    | 00000016 | 03B00 |              | 0. 0. 0. 40. 0. 47. 47. 97. 0. 1. 0. 40. -  |
|   | 00000016 | 03B0F |              | 0. 37. 37. 83. 0. 0. 0. 40. 0. 8. 8. 97. -  |
|   | 00000016 | 03B14 | P.AHK: .LONG | 0. 0. 0. 40. 0. 4. 4. 97. 0. 0. 0. 40. 0. - |
|   | 00000016 | 03B18 | .BYTE        | 10. 10. 98. 0. 0. 0. 40. 0. 11. 11. 99. -   |
|   | 00000016 | 03B27 |              | 0. 0. 0. 40. 0. 27. 27. 100. 0. 0. 0. 40. - |
|   | 00000016 | 03B36 |              | 0. 28. 28. 101. 0. 0. 0. 40. 0              |
| 28 00 00 00 6C 28 28 00 28 00 02 00 D8 33 33 03B45    | 00000016 | 03B45 | P.AHL: .LONG | 26  |
| 00 00 00 5D 25 25 00 28 00 01 00 6C 2F 2F 00 03B54    | 00000016 | 03B54 | .BYTE        | 51. 51. -40. 0. 2. 0. 40. 0. 40. 40. 108. - |
| 00 00 6C 04 04 00 28 00 00 00 6C 08 08 00 28 00 03B63 | 00000016 | 03B63 |              | 0. 0. 0. 40. 0. 47. 47. 108. 0. 1. 0. 40. - |
| 00 6E 0B 0B 00 28 00 00 00 6D 0A 0A 00 28 00 00 03B7B | 00000016 | 03B7B |              | 0. 37. 37. 83. 0. 0. 0. 40. 0. 8. 8. 108. - |
| 70 1C 1C 00 28 00 00 00 6F 1B 1B 00 28 00 00 03B8A    | 00000016 | 03B8A |              | 0. 0. 0. 40. 0. 4. 4. 108. 0. 0. 0. 40. -   |
|   | 00000016 | 03B99 |              | 0. 10. 10. 109. 0. 0. 0. 40. 0. 11. 11. -   |
|   | 00000016 | 03BA8 |              | 110. 0. 0. 0. 40. 0. 27. 27. 111. 0. 0. -   |
|   | 00000016 | 03BB7 |              | 0. 40. 0. 28. 28. 112. 0. 0. 0. 40. 0       |
|   | 00000016 | 03BB8 | P.AHL: .LONG | 26  |
|   | 00000016 | 03BB9 | .BYTE        | 51. 51. -44. 0. 2. 0. 40. 0. 40. 40. 119. - |
|   | 00000016 | 03BBB |              | 0. 0. 0. 40. 0. 47. 47. 119. 0. 1. 0. 40. - |
|   | 00000016 | 03BBB |              | 0. 37. 37. 85. 0. 0. 0. 40. 0. 8. 8. 119. - |
|   | 00000016 | 03BBB |              | 0. 0. 0. 40. 0. 4. 4. 120. 0. 0. 0. 40. -   |
|   | 00000016 | 03BBB |              | 0. 10. 10. 121. 0. 0. 0. 40. 0. 11. 11. -   |
|   | 00000016 | 03BBB |              | 122. 0. 0. 0. 40. 0. 27. 27. 123. 0. 0. -   |
|   | 00000016 | 03BBB |              | 0. 40. 0. 28. 28. 124. 0. 0. 0. 40. 0       |

|   |              |   |
|---|--------------|---|
| 28 00 01 00 7F 2F 2F 00 28 00 00 00 7F 28 28 03BC0    | P.AHM: .BYTE | 40. 40. 127. 0. 0. 0. 40. 0. 47. 47. 127. - |
| 00 00 00 00 7F 08 08 00 28 00 00 00 25 25 00 03BCF    |              | 0. 1. 0. 40. 0. 37. 37. 0. 0. 0. 40. -      |
| 00 00 81 0A 0A 00 00 28 00 00 00 80 04 04 00 03BDE    |              | 0. 8. 8. 127. 0. 0. 0. 40. 4. 4. -128. -    |
| 00 83 1B 1B 00 28 00 00 00 00 82 08 08 00 28 00 03BED |              | 0. 0. 0. 40. 0. 10. 10. -127. 0. 0. 0. -    |
| 00 00 00 00 28 00 00 00 84 1C 1C 00 28 00 00 03BFC    |              | 40. 0. 11. 11. -128. 0. 0. 40. 0. 27. -     |
|   |              | 27. -125. 0. 0. 0. 40. 0. 28. 28. -124. -   |
|   |              | 0. 0. 0. 40. 0                              |
| 28 00 01 00 8F 2F 2F 00 28 00 00 00 8F 28 28 03C08    | P.AHM: .LONG | 18  |
| 00 00 00 00 8F 08 08 00 28 00 00 00 5B 25 25 00 03C0C | P.AHM: .BYTE | 40. 40. -113. 0. 0. 0. 40. 0. 47. 47. -     |
| 00 00 91 0A 0A 00 28 00 00 00 90 04 04 00 28 00 03C1B |              | -113. 0. 1. 0. 40. 0. 37. 37. 91. 0. 0. -   |
| 00 93 1B 1B 00 28 00 00 00 00 92 08 08 00 28 00 03C2A |              | 0. 40. 0. 8. 8. -113. 0. 0. 0. 40. 0. 4. -  |
| 00 00 00 00 28 00 00 00 94 1C 1C 00 28 00 00 03C39    |              | 4. -112. 0. 0. 0. 40. 0. 10. 10. -111. 0. - |
|   |              | 0. 0. 40. 0. 11. 11. -110. 0. 0. 0. 40. -   |
|   |              | 0. 27. 27. -109. 0. 0. 0. 40. 0. 28. 28. -  |
|   |              | -108. 0. 0. 0. 40. 0                        |
| 28 00 00 00 87 28 28 00 28 00 02 00 07 33 33 03C54    | P.AHO: .LONG | 20  |
| 00 00 00 00 59 25 25 00 28 00 01 00 87 2F 2F 00 03C58 | P.AHO: .BYTE | 51. 51. -41. 0. 2. 0. 40. 0. 40. 40. -      |
| 00 00 88 04 04 00 28 00 00 00 87 08 08 00 28 00 03C67 |              | -121. 0. 0. 0. 40. 0. 47. 47. -121. 0. 1. - |
| 00 8A 08 08 00 28 00 00 00 89 0A 0A 00 28 00 03C76    |              | 0. 40. 0. 37. 37. 89. 0. 0. 0. 40. 0. 8. -  |
| 8C 1C 1C 00 28 00 00 00 8B 1B 1B 00 28 00 00 03C85    |              | 8. -121. 0. 0. 0. 40. 0. 4. 4. -120. 0. -   |
|   |              | 0. 0. 40. 0. 10. 10. -119. 0. 0. 0. 40. -   |
|   |              | 0. 11. 11. -118. 0. 0. 0. 40. 0. 27. 27. -  |
|   |              | -117. 0. 0. 0. 40. 0. 28. 28. -116. 0. 0. - |
|   |              | 0. 40. 0                                    |
|   |              | 2   |
|   |              | 40. 40. -86. 0. 0. 0. 40. 0                 |
| 00 28 00 00 00 AA 28 28 03CAC                         | P.AHP: .BYTE |   |
|   |              | 2   |
| 00 28 00 00 00 A6 28 28 03CB4                         | P.AHQ: .LONG |   |
|   |              | 40. 40. -90. 0. 0. 0. 40. 0                 |
| 00 28 00 00 00 AE 28 28 03CB8                         | P.AHQ: .BYTE |   |
|   |              | 2   |
| 00 28 00 00 00 AE 28 28 03CC0                         | P.AHR: .LONG |   |
|   |              | 40. 40. -82. 0. 0. 0. 40. 0                 |
| 28 00 02 00 D5 04 33 00 28 00 02 00 D5 08 33 03CC4    | P.AHS: .BYTE | 10  |
| 00 02 00 D5 28 33 00 28 00 02 00 D5 0E 33 00 03CC8    | P.AHS: .LONG |   |
|   |              | 51. 8. -43. 0. 2. 0. 40. 0. 51. 4. -43. -   |
|   |              | 0. 2. 0. 40. 0. 51. 14. -43. 0. 2. 0. 40. - |
|   |              | 0. 51. 40. -43. 0. 2. 0. 40. 0. 51. 47. -   |
|   |              | -43. 0. 2. 0. 40. 0                         |
|   |              | 2   |
| 00 2F 00 01 01 0E 2F 2F 03CF8                         | P.AHT: .LONG |   |
|   |              | 47. 47. 14. 1. 1. 0. 47. 0                  |
| 00 2F 00 01 01 0F 2F 2F 03D04                         | P.AHU: .LONG |   |
|   |              | 2   |
|   |              | 47. 47. 15. 1. 1. 0. 47. 0                  |
|   |              | 00# 03D10 PASCAL_OPINFO_TABLE:              |
|   |              | .BYTE 0[64]                                 |
| 000038B4 0000386C 000039D0 03D50                      |              | .LONG 14800. 14444. 14516                   |
|   |              | 01 03D5C                                    |
|   |              | .BYTE 1                                     |
| 000038B4 0000386C 00003A2C 03D5D                      |              | .BYTE 0[3]                                  |
|   |              | .LONG 14892. 14444. 14516                   |
|   |              | 01 03D6C                                    |
|   |              | .BYTE 1                                     |
| 000038B4 0000386C 000038BC 03D6D                      |              | .BYTE 0[3]                                  |
|   |              | .LONG 14524. 14444. 14516                   |
|   |              | 01 03D7C                                    |
|   |              | .BYTE 1                                     |
| 000038B4 0000386C 00003900 03D7D                      |              | .BYTE 0[3]                                  |
|   |              | .LONG 14592. 14444. 14516                   |
|   |              | 01 03D8C                                    |
|   |              | .BYTE 1                                     |
| 000038B4 0000386C 0000393C 03D8D                      |              | .BYTE 0[3]                                  |
|   |              | .LONG 14652. 14444. 14516                   |
|   |              | 01 03D9C                                    |
|   |              | .BYTE 1                                     |
| 000038B4 0000386C 000039AC 03D9D                      |              | .BYTE 0[3]                                  |
|   |              | .LONG 14764. 14444. 14516                   |

|          |          |          |       |       |                     |  |
|----------|----------|----------|-------|-------|---------------------|--|
|          |          | 01       | 03DAC | .BYTE | 1                   |  |
|          |          | 00#      | 03DAD | .BYTE | 0[3]                |  |
| 000038B4 | 0000386C | 00003978 | 03DB0 | .LONG | 14712, 14444, 14516 |  |
|          |          | 01       | 03DBC | .BYTE | 1                   |  |
|          |          | 00#      | 03DBD | .BYTE | 0[35]               |  |
| 000038B4 | 0000386C | 00003AC4 | 03DE0 | .LONG | 15044, 14444, 14516 |  |
|          |          | 01       | 03DEC | .BYTE | 1                   |  |
|          |          | 00#      | 03DED | .BYTE | 0[3]                |  |
| 000038B4 | 0000386C | 00003B18 | 03DF0 | .LONG | 15128, 14444, 14516 |  |
|          |          | 01       | 03DFC | .BYTE | 1                   |  |
|          |          | 00#      | 03DFD | .BYTE | 0[3]                |  |
| 000038B4 | 0000386C | 00003BC0 | 03E00 | .LONG | 15296, 14444, 14516 |  |
|          |          | 01       | 03E0C | .BYTE | 1                   |  |
|          |          | 00#      | 03E0D | .BYTE | 0[19]               |  |
| 000038B4 | 0000386C | 00003B6C | 03E20 | .LONG | 15212, 14444, 14516 |  |
|          |          | 01       | 03E2C | .BYTE | 1                   |  |
|          |          | 00#      | 03E2D | .BYTE | 0[19]               |  |
| 000038B4 | 0000386C | 00003C0C | 03E40 | .LONG | 15372, 14444, 14516 |  |
|          |          | 01       | 03E4C | .BYTE | 1                   |  |
|          |          | 00#      | 03E4D | .BYTE | 0[19]               |  |
| 000038B4 | 0000386C | 00003C58 | 03E60 | .LONG | 15448, 14444, 14516 |  |
|          |          | 01       | 03E6C | .BYTE | 1                   |  |
|          |          | 00#      | 03E6D | .BYTE | 0[19]               |  |
| 00000000 | 00000000 | 00003CAC | 03E80 | .LONG | 15532, 0, 0         |  |
|          |          | 01       | 03E8C | .BYTE | 1                   |  |
|          |          | 00#      | 03E8D | .BYTE | 0[3]                |  |
| 00000000 | 00000000 | 00003CB8 | 03E90 | .LONG | 15544, 0, 0         |  |
|          |          | 01       | 03E9C | .BYTE | 1                   |  |
|          |          | 00#      | 03E9D | .BYTE | 0[3]                |  |
| 00000000 | 00000000 | 00003CC4 | 03EA0 | .LONG | 15556, 0, 0         |  |
|          |          | 01       | 03EAC | .BYTE | 1                   |  |
|          |          | 00#      | 03EAD | .BYTE | 0[163]              |  |
| 00000000 | 000038B4 | 00003A9C | 03F50 | .LONG | 15004, 14468, 0     |  |
|          |          | 01       | 03F5C | .BYTE | 1                   |  |
|          |          | 00#      | 03F5D | .BYTE | 0[3]                |  |
| 000038B4 | 000038B4 | 00003AB0 | 03F60 | .LONG | 15024, 14468, 14516 |  |
|          |          | 01       | 03F6C | .BYTE | 1                   |  |
|          |          | 00#      | 03F6D | .BYTE | 0[67]               |  |
| 00000000 | 00000000 | 00003CD0 | 03FB0 | .LONG | 15568, 0, 0         |  |
|          |          | 01       | 03FBC | .BYTE | 1                   |  |
|          |          | 00#      | 03FBD | .BYTE | 0[35]               |  |
| 00000000 | 00000000 | 00003CFC | 03FE0 | .LONG | 15612, 0, 0         |  |
|          |          | 01       | 03FEC | .BYTE | 1                   |  |
|          |          | 00#      | 03FED | .BYTE | 0[3]                |  |
| 00000000 | 00000000 | 00003D08 | 03FF0 | .LONG | 15624, 0, 0         |  |
|          |          | 01       | 03FFC | .BYTE | 1                   |  |
|          |          | 00#      | 03FFD | .BYTE | 0[19]               |  |
| 00000000 | 000038B4 | 00003A88 | 04010 | .LONG | 14984, 14468, 0     |  |
|          |          | 01       | 0401C | .BYTE | 1                   |  |
|          |          | 00#      | 0401D | .BYTE | 0[19]               |  |
| 000038B4 | 00003890 | 00000000 | 04030 | .LONG | 0, 14480, 14516     |  |
|          |          | 01       | 0403C | .BYTE | 1                   |  |
|          |          | 00#      | 0403D | .BYTE | 0[3]                |  |
| 000038B4 | 00003890 | 00000000 | 04040 | .LONG | 0, 14480, 14516     |  |
|          |          | 01       | 0404C | .BYTE | 1                   |  |
|          |          | 00#      | 0404D | .BYTE | 0[67]               |  |
| 00000000 | 00000000 | 00000000 | 04090 | .LONG | 0, 0, 0             |  |

| Address  | Hex Data   | Label              | Value   |
|--|--|--------------------|---|
| 0001FF0E   | 00010EFF 0001FF22 000122FF 0001FF01 0001FF25 00004168                      | P.AHV:             | 179   |
| 0001FF0E   | 00010EFF 0001FF22 000122FF 0001FF01 0001FF25 00004168                      | P.AHW:             | 2107, 296, 0                                      |
| 0001FF0E   | 00010EFF 0001FF22 000122FF 0001FF01 0001FF25 00004168                      | P.AHX:             | 6923, 0   |
| 0001FF0E   | 00010EFF 0001FF22 000122FF 0001FF01 0001FF25 00004168                      | PLI_CVTINFO TABLE: | 66047, 130817, 74495, 130850, 69375, -            |
| 0001FF0E   | 00010EFF 0001FF22 000122FF 0001FF01 0001FF25 00004168                      | PLI_CVTINFO TABLE: | 130830, 75263, 130853, 130864, 78079, 0           |
| 0001FF0E   | 00010EFF 0001FF22 000122FF 0001FF01 0001FF25 00004168                      | PLI_CVTINFO TABLE: | 16724, 16744                                      |
| 2B08 0A08 0B15 1507 0706 1530 1522 1501 1525 150E 0000 1C1B 1C0B 1C2B 1B0A 0B0A        | 00000008 0419C 0419D 041A0 041A4 041B8                                     | P.AHY:             | 5390, 5413, 5377, 5410, 5424, 1798, 5383, -       |
| 2501 0E01 1501 0122 2522 0E22 1522 150E 0E25 1525 0706 1530 0000 1C1B                  | 00000008 041C4 041C8 041DC 041F0   | P.AHZ:             | 2069, 2568, 11016, 2826, 6922, 7211, -            |
| 010E 0130 011C 011B 010B 010A 0115 0108 0107 0106 0000 0122 0125                       | 00000007 041F4 041F8 0420C 04212 04214 04218 0422C 04232 04234 04238 0424C | P.AIA:             | 7179, 7195, 0                                     |
| 0E25 0E30 0E1C 0E1B 0E0B 0E0A 0E15 0E08 0E07 0E06 0000 0E01 0122                       | 00000007 04214 04218 0422C 04232 04234 04238 0424C                         | P.AIB:             | 5413, 3621, 5390, 5410, 3618, 9506, 290, -        |
| 2B08 0A08 0B15 1507 0706 1530 0822 0801 1525 150E 0000 1C1B 1C0B 1C2B 1B0A 0B0A        | 00000008 04234 04238 0424C   | P.AIC:             | 5377, 3585, 9473, 5424, 1798, 5383, 2069, -       |
| 2201 011C 1C1B 1B0B 0B0A 0A2B 2B15 1508 0807 0706 0000 0630 3025 250E 0E22             | 00000008 04258 0425C 04270   | P.AID:             | 2568, 11016, 2826, 6922, 7211, 7179, -            |
| 07 00 00 00 03 07 07 00 06 00 00 00 01 06 06 0427A 0427C 04280 0428F 0429E 042AD 042BC | 00000012 0427A 0427C 04280 0428F 0429E 042AD 042BC                         | P.AIE:             | 7, 6, 1, 0, 0, 0, 6, 0, 7, 7, 3, 0, 0, 0, -       |
| 00 00 00 07 0A 0A 00 08 00 00 00 05 08 08 00 04280 0428F 0429E 042AD 042BC             | 00000012 0427A 0427C 04280 0428F 0429E 042AD 042BC                         | P.AIF:             | 0, 0, 0, 10, 0, 0, 11, 11, 8, 0, 0, 0, 11, -      |
| 00 00 09 1B 1B 00 08 00 00 00 08 08 08 00 0A 04280 0428F 0429E 042AD 042BC             | 00000012 0427A 0427C 04280 0428F 0429E 042AD 042BC                         | P.AIG:             | 0, 0, 0, 27, 27, 4, 0, 0, 0, 27, 0, 28, 28, 10, - |
| 00 F2 15 15 00 1C 00 00 00 0A 1C 1C 00 1B 00 04280 0428F 0429E 042AD 042BC             | 00000012 0427A 0427C 04280 0428F 0429E 042AD 042BC                         | P.AIG:             | 0, 0, 0, 28, 28, 0, 21, 21, -14, 0, 0, 0, 21, -   |
| 00 00 00 00 00 00 01 13 2B 2B 00 15 00 00 04280 0428F 0429E 042AD 042BC                | 00000012 0427A 0427C 04280 0428F 0429E 042AD 042BC                         | P.AIG:             | 0, 43, 43, 19, 1, 0, 0, 43, 0                     |
| 07 00 00 00 11 07 07 00 06 00 00 00 0F 06 06 042C8 042CC 042DB 042EA 042F9 04308       | 00000012 042C8 042CC 042DB 042EA 042F9 04308                               | P.AIG:             | 6, 6, 15, 0, 0, 0, 6, 0, 7, 7, 17, 0, 0, -        |
| 00 00 00 15 0A 0A 00 08 00 00 00 13 08 08 00 042C8 042CC 042DB 042EA 042F9 04308       | 00000012 042C8 042CC 042DB 042EA 042F9 04308                               | P.AIG:             | 0, 7, 0, 8, 8, 19, 0, 0, 0, 8, 0, 10, 10, -       |
| 00 00 17 1B 1B 00 08 00 00 00 16 0B 0B 00 0A 042C8 042CC 042DB 042EA 042F9 04308       | 00000012 042C8 042CC 042DB 042EA 042F9 04308                               | P.AIG:             | 21, 0, 0, 0, 10, 0, 11, 11, 22, 0, 0, 0, -        |
| 00 F3 15 15 00 1C 00 00 00 1B 1C 1C 00 1B 00 042C8 042CC 042DB 042EA 042F9 04308       | 00000012 042C8 042CC 042DB 042EA 042F9 04308                               | P.AIG:             | 11, 0, 27, 27, 23, 0, 0, 0, 27, 0, 28, 28, -      |
| 00 00 00 31 0A 0A 00 08 00 00 00 2F 08 08 00 04314 04318 04327                         | 00000012 04314 04318 04327   | P.AIG:             | 28, 24, 0, 0, 0, 28, 0, 21, 21, -13, 0, -         |

DBGEVALOP  
V04-000

J 5  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 B11ss-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32:1

Page 136  
(22)

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| 00 | 00 | 33 | 1B | 1B | 00 | 0B | 00 | 00 | 00 | 32 | 0B | 0B | 00 | 0A | 04336 |
| 00 | F4 | 15 | 15 | 00 | 1C | 00 | 00 | 00 | 00 | 34 | 1C | 00 | 1B | 00 | 04345 |
|    |    |    | 00 | 2B | 00 | 00 | 01 | 15 | 2B | 2B | 00 | 15 | 00 | 00 | 04354 |
| 07 | 00 | 00 | 00 | 1F | 07 | 07 | 00 | 06 | 00 | 00 | 00 | 1D | 06 | 06 | 04360 |
| 00 | 00 | 00 | 23 | 0A | 0A | 00 | 0B | 00 | 00 | 00 | 21 | 0B | 0B | 00 | 04364 |
| 00 | 00 | 25 | 1B | 1B | 00 | 0B | 00 | 00 | 00 | 24 | 0B | 0B | 00 | 0A | 04373 |
| 00 | F5 | 15 | 15 | 00 | 1C | 00 | 00 | 00 | 00 | 26 | 1C | 1C | 00 | 1B | 04382 |
|    |    |    | 00 | 2B | 00 | 00 | 01 | 16 | 2B | 2B | 00 | 15 | 00 | 00 | 04391 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 043A0 |
| 1A | 00 | 00 | 01 | 0C | 1A | 1A | 00 | 09 | 00 | 00 | 01 | 0A | 09 | 09 | 043AC |
| 00 | 00 | 00 | BE | 07 | 07 | 00 | 06 | 00 | 00 | 00 | BD | 06 | 06 | 00 | 043B0 |
| 00 | 00 | C0 | 0A | 0A | 00 | 0B | 00 | 00 | 00 | BF | 0B | 0B | 00 | 07 | 043BF |
| 00 | C2 | 1B | 1B | 00 | 0B | 00 | 00 | 00 | 00 | C1 | 0B | 0B | 00 | 0A | 043CE |
| F6 | 15 | 15 | 00 | 1C | 00 | 00 | 00 | C3 | 1C | 1C | 00 | 1B | 00 | 00 | 043DD |
|    |    |    | 00 | 2B | 00 | 00 | 01 | 10 | 2B | 2B | 00 | 15 | 00 | 00 | 043EC |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 043FB |
| 1A | 00 | 00 | 01 | 0D | 1A | 1A | 00 | 09 | 00 | 00 | 01 | 0B | 09 | 09 | 04408 |
| 00 | 00 | 00 | B2 | 07 | 07 | 00 | 06 | 00 | 00 | 00 | BD | 06 | 06 | 00 | 0440C |
| 00 | 00 | B5 | 0A | 0A | 00 | 0B | 00 | 00 | 00 | 00 | B3 | 0B | 00 | 07 | 0441B |
| 00 | B7 | 1B | 1B | 00 | 0B | 00 | 00 | 00 | 00 | B6 | 0B | 0B | 00 | 0A | 0442A |
| F7 | 15 | 15 | 00 | 1C | 00 | 00 | 00 | B8 | 1C | 1C | 00 | 1B | 00 | 00 | 04439 |
|    |    |    | 00 | 2B | 00 | 00 | 01 | 11 | 2B | 2B | 00 | 15 | 00 | 00 | 04448 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04457 |
| 0B | 00 | 00 | 00 | 43 | 0B | 0B | 00 | 0A | 00 | 00 | 00 | 42 | 0A | 0B | 04464 |
| 00 | 00 | 00 | 45 | 1C | 0B | 00 | 1B | 00 | 00 | 00 | 44 | 1B | 0B | 00 | 04468 |
| 00 | 00 | 4A | 0B | 0A | 00 | 0A | 00 | 00 | 00 | 49 | 0A | 0A | 00 | 1C | 04477 |
| 00 | 4C | 0B | 0B | 00 | 0B | 00 | 00 | 00 | 4B | 0A | 0B | 00 | 0B | 00 | 04486 |
| 4E | 1C | 1C | 00 | 1B | 00 | 00 | 00 | 4D | 1B | 1B | 00 | 0B | 00 | 00 | 04495 |
|    |    |    |    |    |    |    |    |    |    |    | 00 | 1C | 00 | 00 | 044A4 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 044B3 |
| 01 | 00 | 00 | 00 | FE | 01 | 01 | 00 | 0E | 00 | 00 | 00 | 52 | 0E | 0E | 044B8 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 044BC |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 044CB |
| 2B | 00 | 00 | 00 | 60 | 07 | 07 | 00 | 2B | 00 | 00 | 00 | 5F | 06 | 06 | 044CC |
| 00 | 00 | 00 | 62 | 0A | 0A | 00 | 2B | 00 | 00 | 00 | 61 | 0B | 0B | 00 | 044D0 |
| 00 | 00 | 64 | 1B | 1B | 00 | 2B | 00 | 00 | 00 | 63 | 0B | 0B | 00 | 2B | 044DF |
| 00 | F8 | 15 | 15 | 00 | 2B | 00 | 00 | 00 | 65 | 1C | 1C | 00 | 2B | 00 | 044EE |
| 54 | 0E | 0E | 00 | 2B | 00 | 00 | 01 | 17 | 2B | 2B | 00 | 2B | 00 | 00 | 044FD |
| 01 | 01 | 00 | 2B | 00 | 00 | 00 | 53 | 25 | 25 | 00 | 2B | 00 | 00 | 00 | 0450C |
|    |    |    | 00 | 00 | 00 | FF | 22 | 22 | 00 | 2B | 00 | 00 | 00 | FF | 0451B |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0452A |
| 2B | 00 | 00 | 00 | 6B | 07 | 07 | 00 | 2B | 00 | 00 | 00 | 6A | 06 | 06 | 04538 |
| 00 | 00 | 00 | 6D | 0A | 0A | 00 | 2B | 00 | 00 | 00 | 6C | 0B | 0B | 00 | 0453C |
| 00 | 00 | 6F | 1B | 1B | 00 | 2B | 00 | 00 | 00 | 6E | 0B | 0B | 00 | 2B | 0454B |
| 00 | F9 | 15 | 15 | 00 | 2B | 00 | 00 | 00 | 70 | 1C | 1C | 00 | 2B | 00 | 0455A |
| 5E | 0E | 0E | 00 | 2B | 00 | 00 | 01 | 1B | 2B | 2B | 00 | 2B | 00 | 00 | 04569 |
| 01 | 01 | 00 | 2B | 00 | 00 | 00 | 5D | 25 | 25 | 00 | 2B | 00 | 00 | 00 | 04578 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04587 |

P.AIH: .LONG  
.BYTE

P.AII: .LONG  
.BYTE

P.AIJ: .LONG  
.BYTE

P.AIK: .LONG  
.BYTE

P.AIL: .LONG  
.BYTE

P.AIM: .LONG  
.BYTE

P.AIN: .LONG  
.BYTE

|     |      |      |      |     |      |      |      |     |      |      |     |    |    |   |
|-----|------|------|------|-----|------|------|------|-----|------|------|-----|----|----|---|
| 49. | 0.   | 0.   | 0.   | 10. | 0.   | 11.  | 11.  | 50. | 0.   | 0.   | 0.  | -  |    |   |
| 11. | 0.   | 27.  | 27.  | 51. | 0.   | 0.   | 0.   | 27. | 0.   | 28.  | 0.  | -  |    |   |
| 28. | 52.  | 0.   | 0.   | 0.  | 28.  | 0.   | 21.  | 21. | -12. | 0.   | -   |    |    |   |
| 0.  | 0.   | 21.  | 0.   | 43. | 43.  | 21.  | 1.   | 0.  | 0.   | 43.  | 0.  |    |    |   |
| 18. | 6.   | 29.  | 0.   | 0.  | 0.   | 6.   | 0.   | 7.  | 7.   | 31.  | 0.  | -  |    |   |
| 0.  | 7.   | 0.   | 8.   | 8.  | 33.  | 0.   | 0.   | 8.  | 0.   | 10.  | 10. | -  |    |   |
| 35. | 0.   | 0.   | 0.   | 10. | 0.   | 11.  | 11.  | 36. | 0.   | 0.   | 0.  | -  |    |   |
| 11. | 0.   | 27.  | 27.  | 37. | 0.   | 0.   | 0.   | 27. | 0.   | 28.  | 0.  | -  |    |   |
| 28. | 38.  | 0.   | 0.   | 0.  | 28.  | 0.   | 21.  | 21. | -11. | 0.   | -   |    |    |   |
| 0.  | 0.   | 21.  | 0.   | 43. | 43.  | 22.  | 1.   | 0.  | 0.   | 43.  | 0.  |    |    |   |
| 22. | 9.   | 10.  | 1.   | 0.  | 0.   | 9.   | 0.   | 26. | 26.  | 12.  | 1.  | -  |    |   |
| 0.  | 0.   | 26.  | 0.   | 6.  | 6.   | -67. | 0.   | 0.  | 0.   | 6.   | 0.  | -  |    |   |
| 7.  | -66. | 0.   | 0.   | 0.  | 7.   | 0.   | 8.   | 8.  | -65. | 0.   | 0.  | -  |    |   |
| 0.  | 8.   | 0.   | 10.  | 10. | -64. | 0.   | 0.   | 0.  | 10.  | 0.   | 11. | -  |    |   |
| 11. | -63. | 0.   | 0.   | 0.  | 11.  | 0.   | 27.  | 27. | -62. | 0.   | -   |    |    |   |
| 0.  | 0.   | 27.  | 0.   | 28. | 28.  | -61. | 0.   | 0.  | 0.   | 28.  | 0.  | -  |    |   |
| 21. | 21.  | -10. | 0.   | 0.  | 0.   | 21.  | 0.   | 43. | 43.  | 16.  | -   |    |    |   |
| 1.  | 0.   | 0.   | 43.  | 0.  |      |      |      |     |      |      |     |    |    |   |
| 22. | 9.   | 11.  | 1.   | 0.  | 0.   | 9.   | 0.   | 26. | 26.  | 13.  | 1.  | -  |    |   |
| 0.  | 0.   | 26.  | 0.   | 6.  | 6.   | -79. | 0.   | 0.  | 0.   | 6.   | 0.  | -  |    |   |
| 7.  | -78. | 0.   | 0.   | 0.  | 7.   | 0.   | 8.   | 8.  | -77. | 0.   | 0.  | -  |    |   |
| 0.  | 8.   | 0.   | 10.  | 10. | -75. | 0.   | 0.   | 0.  | 10.  | 0.   | 11. | -  |    |   |
| 11. | -74. | 0.   | 0.   | 0.  | 11.  | 0.   | 27.  | 27. | -73. | 0.   | -   |    |    |   |
| 0.  | 0.   | 27.  | 0.   | 28. | 28.  | -72. | 0.   | 0.  | 0.   | 28.  | 0.  | -  |    |   |
| 21. | 21.  | -9.  | 0.   | 0.  | 0.   | 21.  | 0.   | 43. | 43.  | 17.  | -   |    |    |   |
| 1.  | 0.   | 0.   | 43.  | 0.  |      |      |      |     |      |      |     |    |    |   |
| 20. | 8.   | 10.  | 66.  | 0.  | 0.   | 0.   | 10.  | 0.  | 8.   | 11.  | 67. | 0. | -  |   |
| 0.  | 0.   | 11.  | 0.   | 8.  | 27.  | 68.  | 0.   | 0.  | 0.   | 27.  | 0.  | -  |    |   |
| 8.  | 28.  | 69.  | 0.   | 0.  | 0.   | 28.  | 0.   | 10. | 10.  | 73.  | 0.  | -  |    |   |
| 0.  | 0.   | 10.  | 0.   | 10. | 11.  | 74.  | 0.   | 0.  | 0.   | 11.  | 0.  | -  |    |   |
| 11. | 10.  | 75.  | 0.   | 0.  | 0.   | 11.  | 0.   | 11. | 11.  | 76.  | -   |    |    |   |
| 0.  | 0.   | 0.   | 11.  | 0.  | 27.  | 27.  | 77.  | 0.  | 0.   | 0.   | 27. | -  |    |   |
| 0.  | 28.  | 28.  | 78.  | 0.  | 0.   | 0.   | 28.  | 0.  |      |      |     |    |    |   |
| 4.  | 14.  | 14.  | 82.  | 0.  | 0.   | 0.   | 14.  | 0.  | 1.   | 1.   | -2. | 0. | -  |   |
| 0.  | 0.   | 1.   | 0.   |     |      |      |      |     |      |      |     |    |    |   |
| 26. | 6.   | 6.   | 95.  | 0.  | 0.   | 0.   | 40.  | 0.  | 7.   | 7.   | 96. | 0. | 0. | - |
| 0.  | 40.  | 0.   | 8.   | 8.  | 97.  | 0.   | 0.   | 0.  | 40.  | 0.   | 10. | 0. | -  |   |
| 10. | 98.  | 0.   | 0.   | 0.  | 40.  | 0.   | 11.  | 11. | 99.  | 0.   | 0.  | -  |    |   |
| 0.  | 40.  | 0.   | 27.  | 27. | 100. | 0.   | 0.   | 0.  | 40.  | 0.   | -   |    |    |   |
| 28. | 28.  | 101. | 0.   | 0.  | 40.  | 0.   | 21.  | 21. | -8.  | -    |     |    |    |   |
| 0.  | 0.   | 0.   | 40.  | 0.  | 43.  | 43.  | 23.  | 1.  | 0.   | 40.  | -   |    |    |   |
| 0.  | 14.  | 14.  | 84.  | 0.  | 0.   | 0.   | 40.  | 0.  | 37.  | 37.  | -   |    |    |   |
| 83. | 0.   | 0.   | 0.   | 40. | 0.   | 1.   | 1.   | -1. | 0.   | 0.   | 0.  | -  |    |   |
| 40. | 0.   | 34.  | 34.  | -1. | 0.   | 0.   | 0.   | 40. | 0.   |      |     |    |    |   |
| 26. | 6.   | 106. | 0.   | 0.  | 0.   | 40.  | 0.   | 7.  | 7.   | 107. | 0.  | -  |    |   |
| 0.  | 0.   | 40.  | 0.   | 8.  | 8.   | 108. | 0.   | 0.  | 0.   | 40.  | 0.  | -  |    |   |
| 10. | 10.  | 109. | 0.   | 0.  | 0.   | 40.  | 0.   | 11. | 11.  | 110. | -   |    |    |   |
| 0.  | 0.   | 0.   | 40.  | 0.  | 27.  | 27.  | 111. | 0.  | 0.   | 0.   | 40. | -  |    |   |
| 0.  | 28.  | 28.  | 112. | 0.  | 0.   | 0.   | 40.  | 0.  | 21.  | 21.  | -   |    |    |   |
| -7. | 0.   | 0.   | 0.   | 40. | 0.   | 43.  | 43.  | 24. | 1.   | 0.   | 0.  | -  |    |   |

00 28 00 00 01 00 22 22 00 28 00 00 01 00 04596  
28 00 00 00 7E 07 07 00 28 00 00 00 7D 06 06 045A4  
00 00 00 81 0A 0A 00 28 00 00 00 7F 08 08 00 045A8  
00 00 83 1B 1B 00 28 00 00 00 82 08 08 00 045B7  
00 FA 15 15 00 28 00 00 00 84 1C 1C 00 28 00 045C6  
58 0E 0E 00 28 00 00 01 1A 28 28 00 28 00 00 045D5  
22 22 00 28 00 00 00 57 25 25 00 28 00 00 00 045E4  
00 28 00 00 01 01 01 01 00 28 00 00 01 01 045F3  
04602

P.AIO: .LONG  
.BYTE

28 00 00 00 76 07 07 00 28 00 00 00 75 06 06 04610  
00 00 00 79 0A 0A 00 28 00 00 00 77 08 08 00 04614  
00 00 7B 1B 1B 00 28 00 00 00 7A 08 08 00 28 04623  
00 FB 15 15 00 28 00 00 00 7C 1C 1C 00 28 00 04632  
56 0E 0E 00 28 00 00 01 1C 28 28 00 28 00 00 04641  
22 22 00 28 00 00 00 55 25 25 00 28 00 00 00 04650  
00 28 00 00 01 02 01 01 00 28 00 00 01 02 0465F  
0466E

P.AIP: .LONG  
.BYTE

28 00 00 00 8E 07 07 00 28 00 00 00 8D 06 06 0467C  
00 00 00 91 0A 0A 00 28 00 00 00 8F 08 08 00 04680  
00 00 93 1B 1B 00 28 00 00 00 92 08 08 00 28 0468F  
00 FC 15 15 00 28 00 00 00 94 1C 1C 00 28 00 0469E  
5C 0E 0E 00 28 00 00 01 19 28 28 00 28 00 00 046AD  
22 22 00 28 00 00 00 58 25 25 00 28 00 00 00 046BC  
00 28 00 00 01 03 01 01 00 28 00 00 01 03 046CB  
046DA

P.AIQ: .LONG  
.BYTE

28 00 00 00 86 07 07 00 28 00 00 00 85 06 06 046E8  
00 00 00 89 0A 0A 00 28 00 00 00 87 08 08 00 046EC  
00 00 8B 1B 1B 00 28 00 00 00 8A 08 08 00 28 046FB  
00 FD 15 15 00 28 00 00 00 8C 1C 1C 00 28 00 0470A  
5A 0E 0E 00 28 00 00 01 1B 28 28 00 28 00 00 04719  
22 22 00 28 00 00 00 59 25 25 00 28 00 00 00 04728  
00 28 00 00 01 04 01 01 00 28 00 00 01 04 04737  
04746

P.AIR: .LONG  
.BYTE

00 01 00 00 01 05 00000002 04754  
00 01 00 00 01 06 00000002 04758  
00 01 00 00 01 07 00000002 04760  
00 01 00 00 01 07 00000002 04764  
00 01 00 00 01 07 00000002 0476C  
00 01 00 00 01 07 00000002 04770  
00 01 00 00 01 07 00000002 04778

P.AIS: .LONG  
.BYTEP.AIT: .LONG  
.BYTEP.AIU: .LONG  
.BYTEPLI\_OPINFO TABLE:  
.BYTE

00004160 00004238 00004380 04788  
01 047C4  
00 047C5  
00004160 00004238 0000440C 047C8  
01 047D4  
00 047D5

.LONG

.BYTE

.LONG

.BYTE

40. 0. 14. 14. 94. 0. 0. 0. 40. 0. 37. -  
37. 93. 0. 0. 0. 40. 0. 1. 1. 0. 0. -  
40. 0. 34. 34. 0. 1. 0. 0. 40. 0. 0. -  
26. 6. 125. 0. 0. 0. 40. 0. 7. 7. 126. 0. -  
0. 0. 40. 0. 8. 8. 127. 0. 0. 0. 40. 0. -  
10. 10. -127. 0. 0. 0. 40. 0. 11. 11. -  
-126. 0. 0. 0. 40. 0. 27. 27. -125. 0. 0. -  
0. 40. 0. 28. 28. -124. 0. 0. 0. 40. 0. -  
21. 21. -6. 0. 0. 0. 40. 0. 43. 43. 26. -  
1. 0. 0. 40. 0. 14. 14. 88. 0. 0. 0. 40. -  
0. 37. 37. 87. 0. 0. 0. 40. 0. 34. 34. 1. -  
1. 0. 0. 40. 0. 1. 1. 1. 1. 0. 0. 40. 0. -  
26. 6. 117. 0. 0. 0. 40. 0. 7. 7. 118. 0. -  
0. 0. 40. 0. 8. 8. 119. 0. 0. 0. 40. 0. -  
10. 10. 121. 0. 0. 0. 40. 0. 11. 11. 122. -  
0. 0. 0. 40. 0. 27. 27. 123. 0. 0. 0. 40. -  
0. 28. 28. 124. 0. 0. 0. 40. 0. 21. 21. -  
-5. 0. 0. 0. 40. 0. 43. 43. 28. 1. 0. 0. -  
40. 0. 14. 14. 86. 0. 0. 0. 40. 0. 37. -  
37. 85. 0. 0. 0. 40. 0. 34. 34. 2. 1. 0. -  
0. 40. 0. 1. 1. 2. 1. 0. 0. 40. 0. -  
26. 6. -115. 0. 0. 0. 40. 0. 7. 7. -114. -  
0. 0. 0. 40. 0. 8. 8. -113. 0. 0. 0. 40. -  
0. 10. 10. -111. 0. 0. 0. 40. 0. 11. 11. -  
-110. 0. 0. 0. 40. 0. 27. 27. -109. 0. 0. -  
0. 40. 0. 28. 28. -108. 0. 0. 0. 40. 0. -  
21. 21. -4. 0. 0. 0. 40. 0. 43. 43. 25. -  
1. 0. 0. 40. 0. 14. 14. 92. 0. 0. 0. 40. -  
0. 37. 37. 91. 0. 0. 0. 40. 0. 34. 34. 3. -  
1. 0. 0. 40. 0. 1. 1. 3. 1. 0. 0. 40. 0. -  
26. 6. -123. 0. 0. 0. 40. 0. 7. 7. -122. -  
0. 0. 0. 40. 0. 8. 8. -121. 0. 0. 0. 40. -  
0. 10. 10. -119. 0. 0. 0. 40. 0. 11. 11. -  
-118. 0. 0. 0. 40. 0. 27. 27. -117. 0. 0. -  
0. 40. 0. 28. 28. -116. 0. 0. 0. 40. 0. -  
21. 21. -3. 0. 0. 0. 40. 0. 43. 43. 27. -  
1. 0. 0. 40. 0. 14. 14. 90. 0. 0. 0. 40. -  
0. 37. 37. 89. 0. 0. 0. 40. 0. 34. 34. 4. -  
1. 0. 0. 40. 0. 1. 1. 4. 1. 0. 0. 40. 0. -  
2. 1. 5. 1. 0. 0. 1. 0  
2. 1. 6. 1. 0. 0. 1. 0  
2. 1. 7. 1. 0. 0. 1. 0  
0[64]  
17328. 16952. 16736  
1  
0[3]  
17420. 16952. 16736  
1  
0[3]

|          |          |          |       |                    |                     |
|----------|----------|----------|-------|--------------------|---------------------|
| 00004160 | 000041A4 | 00004280 | 047D8 | .LONG              | 17024, 16804, 16736 |
|          |          | 01       | 047E4 | .BYTE              | 1                   |
|          |          | 00#      | 047E5 | .BYTE              | 0[3]                |
| 00004160 | 000041A4 | 000042CC | 047E8 | .LONG              | 17100, 16804, 16736 |
|          |          | 01       | 047F4 | .BYTE              | 1                   |
|          |          | 00#      | 047F5 | .BYTE              | 0[3]                |
| 00004160 | 000041A4 | 00004318 | 047F8 | .LONG              | 17176, 16804, 16736 |
|          |          | 01       | 04804 | .BYTE              | 1                   |
|          |          | 00#      | 04805 | .BYTE              | 0[3]                |
| 00004160 | 000041A4 | 00004364 | 04808 | .LONG              | 17252, 16804, 16736 |
|          |          | 01       | 04814 | .BYTE              | 1                   |
|          |          | 00#      | 04815 | .BYTE              | 0[3]                |
| 00004160 | 000041A4 | 00004468 | 04818 | .LONG              | 17512, 16804, 16736 |
|          |          | 01       | 04824 | .BYTE              | 1                   |
|          |          | 00#      | 04825 | .BYTE              | 0[35]               |
| 00004160 | 000041C8 | 000044D0 | 04848 | .LONG              | 17616, 16840, 16736 |
|          |          | 01       | 04854 | .BYTE              | 1                   |
|          |          | 00#      | 04855 | .BYTE              | 0[3]                |
| 00004160 | 000041C8 | 0000453C | 04858 | .LONG              | 17724, 16840, 16736 |
|          |          | 01       | 04864 | .BYTE              | 1                   |
|          |          | 00#      | 04865 | .BYTE              | 0[3]                |
| 00004160 | 000041C8 | 000045A8 | 04868 | .LONG              | 17832, 16840, 16736 |
|          |          | 01       | 04874 | .BYTE              | 1                   |
|          |          | 00#      | 04875 | .BYTE              | 0[19]               |
| 00004160 | 000041C8 | 00004614 | 04888 | .LONG              | 17940, 16840, 16736 |
|          |          | 01       | 04894 | .BYTE              | 1                   |
|          |          | 00#      | 04895 | .BYTE              | 0[19]               |
| 00004160 | 000041C8 | 00004680 | 048A8 | .LONG              | 18048, 16840, 16736 |
|          |          | 01       | 048B4 | .BYTE              | 1                   |
|          |          | 00#      | 048B5 | .BYTE              | 0[19]               |
| 00004160 | 000041C8 | 000046EC | 048C8 | .LONG              | 18156, 16840, 16736 |
|          |          | 01       | 048D4 | .BYTE              | 1                   |
|          |          | 00#      | 048D5 | .BYTE              | 0[131]              |
| 00004160 | 000041F8 | 00004758 | 04958 | .LONG              | 18264, 16888, 16736 |
|          |          | 01       | 04964 | .BYTE              | 1                   |
|          |          | 00#      | 04965 | .BYTE              | 0[3]                |
| 00004160 | 000041F8 | 00004764 | 04968 | .LONG              | 18276, 16888, 16736 |
|          |          | 01       | 04974 | .BYTE              | 1                   |
|          |          | 00#      | 04975 | .BYTE              | 0[3]                |
| 00004160 | 000041F8 | 00004770 | 04978 | .LONG              | 18288, 16888, 16736 |
|          |          | 01       | 04984 | .BYTE              | 1                   |
|          |          | 00#      | 04985 | .BYTE              | 0[35]               |
| 00004160 | 00004218 | 000044BC | 049A8 | .LONG              | 17596, 16920, 16736 |
|          |          | 01       | 049B4 | .BYTE              | 1                   |
|          |          | 00#      | 049B5 | .BYTE              | 0[227]              |
| 00004160 | 0000425C | 00000000 | 04A98 | .LONG              | 0, 16988, 16736     |
|          |          | 01       | 04AA4 | .BYTE              | 1                   |
|          |          | 00#      | 04AA5 | .BYTE              | 0[3]                |
| 00000000 | 0000425C | 00000000 | 04AAB | .LONG              | 0, 16988, 0         |
|          |          | 01       | 04AB4 | .BYTE              | 1                   |
|          |          | 00#      | 04AB5 | .BYTE              | 0[67]               |
| 00000000 | 00000000 | 00000000 | 04AF8 | .LONG              | 0, 0, 0             |
|          |          | 01       | 04B04 | .BYTE              | 1                   |
|          |          |          | 04B05 | .BLKB              | 179                 |
|          | 00000000 | 00000000 | 04BB8 | RPG_CVTINFO TABLE: |                     |
|          |          | 00       | 04BC0 | .LONG              | 0, 0                |
|          |          |          |       | .BYTE              | 0                   |

| 07 | 00 | 00 | 00 | BE | 07 | 07 | 00 | 06 | 00 | 00 | 00 | BD | 06 | 06 | 04BC1 | .BLKB  | 3   |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|--------|---|
| 00 | 00 | 00 | C0 | 0A | 0A | 00 | 08 | 00 | 00 | 00 | BF | 08 | 08 | 00 | 04BC4 | .LONG  | 383, 5384, 5395, 0                          |
| 00 | 00 | C2 | 1B | 1B | 00 | 08 | 00 | 00 | 00 | C1 | 0B | 0B | 00 | 0A | 04BC8 | P.AIV: | .WORD                                       |
| 01 | 0A | 09 | 09 | 00 | 1C | 00 | 00 | 00 | 00 | C3 | 1C | 1C | 00 | 1B | 04BD0 | .LONG  | 2055, 4872, 5395, 1813, 0                   |
| F6 | 15 | 15 | 00 | 1A | 00 | 00 | 01 | 0C | 1A | 1A | 00 | 09 | 00 | 00 | 04BD4 | P.AIW: | .WORD                                       |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04BDE | .BLKB  |   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04BE0 | .LONG  |   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04BE4 | P.AIX: | .BYTE                                       |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04BEC | .LONG  | 21, 21, -14, 0, 0, 0, 21, 0                 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04BF0 | P.AIY: | .BYTE                                       |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04BF8 | .LONG  | 21, 21, -13, 0, 0, 0, 21, 0                 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04BFC | P.AIZ: | .BYTE                                       |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04C04 | .LONG  | 21, 21, -12, 0, 0, 0, 21, 0                 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04C08 | P.AJA: | .BYTE                                       |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04C10 | .LONG  | 21, 21, -11, 0, 0, 0, 21, 0                 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04C14 | P.AJB: | .BYTE                                       |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04C23 |        | 6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04C32 |        | 0, 0, 7, 0, 8, 8, -65, 0, 0, 0, 8, 0, 10, - |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04C41 |        | 10, -64, 0, 0, 0, 10, 0, 11, 11, -63, 0, -  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04C50 |        | 0, 0, 11, 0, 27, 27, -62, 0, 0, 0, 27, 0, - |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 04C5F |        | 28, 28, -61, 0, 0, 0, 28, 0, 9, 9, 10, 1, - |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 0, 0, 9, 0, 26, 26, 12, 1, 0, 0, 26, 0, -   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 21, 21, -10, 0, 0, 0, 21, 0                 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 20  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 0, 0, 7, 0, 8, 8, -77, 0, 0, 0, 8, 0, 10, - |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 10, -75, 0, 0, 0, 10, 0, 11, 11, -74, 0, -  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 0, 0, 11, 0, 27, 27, -73, 0, 0, 0, 27, 0, - |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 28, 28, -72, 0, 0, 0, 28, 0, 9, 9, 11, 1, - |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 0, 0, 9, 0, 26, 26, 13, 1, 0, 0, 26, 0, -   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 21, 21, -9, 0, 0, 0, 21, 0                  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 4   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 14, 14, 84, 0, 0, 0, 40, 0, 21, 21, -8, -   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 0, 0, 0, 40, 0                              |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 4   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        | 14, 14, 94, 0, 0, 0, 40, 0, 21, 21, -7, -   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        |   |

|          |          |          |       |                        |                 |  |
|----------|----------|----------|-------|------------------------|-----------------|--|
|          |          | 01       | 04DA0 | .BYTE                  | 1               |  |
|          |          | 00#      | 04DA1 | .BYTE                  | 0[3]            |  |
| 00000000 | 00004BC8 | 00004C68 | 04DA4 | .LONG                  | 19560, 19400, 0 |  |
|          |          | 01       | 04DB0 | .BYTE                  | 1               |  |
|          |          | 00#      | 04DB1 | .BYTE                  | 0[3]            |  |
| 00000000 | 00004BC8 | 00004BE4 | 04DB4 | .LONG                  | 19428, 19400, 0 |  |
|          |          | 01       | 04DC0 | .BYTE                  | 1               |  |
|          |          | 00#      | 04DC1 | .BYTE                  | 0[3]            |  |
| 00000000 | 00004BC8 | 00004BF0 | 04DC4 | .LONG                  | 19440, 19400, 0 |  |
|          |          | 01       | 04DD0 | .BYTE                  | 1               |  |
|          |          | 00#      | 04DD1 | .BYTE                  | 0[3]            |  |
| 00000000 | 00004BC8 | 00004BFC | 04DD4 | .LONG                  | 19452, 19400, 0 |  |
|          |          | 01       | 04DE0 | .BYTE                  | 1               |  |
|          |          | 00#      | 04DE1 | .BYTE                  | 0[3]            |  |
| 00000000 | 00004BC8 | 00004C08 | 04DE4 | .LONG                  | 19464, 19400, 0 |  |
|          |          | 01       | 04DF0 | .BYTE                  | 1               |  |
|          |          | 00#      | 04DF1 | .BYTE                  | 0[51]           |  |
| 00000000 | 00004BC8 | 00004CBC | 04E24 | .LONG                  | 19644, 19400, 0 |  |
|          |          | 01       | 04E30 | .BYTE                  | 1               |  |
|          |          | 00#      | 04E31 | .BYTE                  | 0[3]            |  |
| 00000000 | 00004BC8 | 00004CD0 | 04E34 | .LONG                  | 19664, 19400, 0 |  |
|          |          | 01       | 04E40 | .BYTE                  | 1               |  |
|          |          | 00#      | 04E41 | .BYTE                  | 0[3]            |  |
| 00000000 | 00004BC8 | 00004CE4 | 04E44 | .LONG                  | 19684, 19400, 0 |  |
|          |          | 01       | 04E50 | .BYTE                  | 1               |  |
|          |          | 00#      | 04E51 | .BYTE                  | 0[19]           |  |
| 00000000 | 00004BC8 | 00004CF8 | 04E64 | .LONG                  | 19704, 19400, 0 |  |
|          |          | 01       | 04E70 | .BYTE                  | 1               |  |
|          |          | 00#      | 04E71 | .BYTE                  | 0[19]           |  |
| 00000000 | 00004BC8 | 00004D0C | 04E84 | .LONG                  | 19724, 19400, 0 |  |
|          |          | 01       | 04E90 | .BYTE                  | 1               |  |
|          |          | 00#      | 04E91 | .BYTE                  | 0[19]           |  |
| 00000000 | 00004BC8 | 00004D20 | 04EA4 | .LONG                  | 19744, 19400, 0 |  |
|          |          | 01       | 04EB0 | .BYTE                  | 1               |  |
|          |          | 00#      | 04EB1 | .BYTE                  | 0[19]           |  |
| 00000000 | 00004BC8 | 00004D34 | 04EC4 | .LONG                  | 19764, 19400, 0 |  |
|          |          | 01       | 04ED0 | .BYTE                  | 1               |  |
|          |          | 00#      | 04ED1 | .BYTE                  | 0[3]            |  |
| 00000000 | 00004BC8 | 00004D40 | 04ED4 | .LONG                  | 19776, 19400, 0 |  |
|          |          | 01       | 04EE0 | .BYTE                  | 1               |  |
|          |          | 00#      | 04EE1 | .BYTE                  | 0[3]            |  |
| 00000000 | 00004BC8 | 00004D4C | 04EE4 | .LONG                  | 19788, 19400, 0 |  |
|          |          | 01       | 04EF0 | .BYTE                  | 1               |  |
|          |          | 00#      | 04EF1 | .BYTE                  | 0[387]          |  |
| 00000000 | 00004BD4 | 00000000 | 05074 | .LONG                  | 0, 19412, 0     |  |
|          |          | 01       | 05080 | .BYTE                  | 1               |  |
|          |          | 00#      | 05081 | .BYTE                  | 0[3]            |  |
| 00000000 | 00004BD4 | 00000000 | 05084 | .LONG                  | 0, 19412, 0     |  |
|          |          | 01       | 05090 | .BYTE                  | 1               |  |
|          |          | 00#      | 05091 | .BYTE                  | 0[67]           |  |
| 00000000 | 00000000 | 00000000 | 050D4 | .LONG                  | 0, 0, 0         |  |
|          |          | 01       | 050E0 | .BYTE                  | 1               |  |
|          |          |          | 050E1 | .BLKB                  | 179             |  |
|          | 00000000 | 00000000 | 05194 | UNKNOWN_CVTINFO_TABLE: |                 |  |
|          |          |          |       | .LONG                  | 0, 0            |  |
|          |          | 00       | 0519C | .BYTE                  | 0               |  |
|          |          |          | 0519D | .BLKB                  | 3               |  |

|      |      |      |      |      |      |      |      |      |      |          |              |
|------|------|------|------|------|------|------|------|------|------|----------|--------------|
| 0A08 | 0807 | 0706 | 0804 | 0403 | 0302 | 082F | 0831 | 083B | 0828 | 0000000A | 051A0        |
| 0000 | 1D0C | 0D0C | 1D1B | 1C1B | 0D0B | 1C0B | 0C0A | 1B0A | 0B0A | 051A4    | P.AJM: .LONG |
|      |      |      |      |      |      |      |      |      |      | 051B8    | .WORD        |
| 1C1B | 1C0B | 1B0A | 0B0A | 0A08 | 0804 | 0407 | 0703 | 0306 | 0602 | 0000000A | 051CC        |
| 0000 | 0228 | 283B | 3B31 | 312F | 2F1D | 2F0D | 1D0C | 0D0C | 0C1C | 051D0    | P.AJN: .LONG |
|      |      |      |      |      |      |      |      |      |      | 051E4    | .WORD        |
|      |      |      |      |      | 0000 | 1D0D | 0D1B | 1D0B | 1B0B | 00000003 | 051F8        |
|      |      |      |      |      |      |      |      |      |      | 051FC    | P.AJO: .LONG |
|      |      |      |      |      |      |      |      |      |      | 05206    | .WORD        |
| 0A   | 00   | 00   | 00   | 07   | 0A   | 0A   | 00   | 08   | 00   | 00       | 00           |
| 00   | 00   | 00   | 09   | 1B   | 1B   | 00   | 08   | 00   | 00   | 00       | 08           |
| 00   | 00   | 0B   | 0C   | 0C   | 00   | 1C   | 00   | 00   | 00   | 0A       | 1C           |
| 00   | 0D   | 1D   | 1D   | 00   | 0D   | 00   | 00   | 00   | 0C   | 0D       | 0D           |
|      |      |      |      |      |      |      |      |      |      | 00       | 1D           |
|      |      |      |      |      |      |      |      |      |      | 00       | 00           |
| 0A   | 00   | 00   | 00   | 15   | 0A   | 0A   | 00   | 08   | 00   | 00       | 00           |
| 00   | 00   | 00   | 17   | 1B   | 1B   | 00   | 08   | 00   | 00   | 00       | 13           |
| 00   | 00   | 19   | 0C   | 0C   | 00   | 1C   | 00   | 00   | 00   | 1B       | 0B           |
| 00   | 1B   | 1D   | 1D   | 00   | 0D   | 00   | 00   | 00   | 1A   | 0D       | 0D           |
|      |      |      |      |      |      |      |      |      |      | 00       | 1D           |
|      |      |      |      |      |      |      |      |      |      | 00       | 00           |
| 0A   | 00   | 00   | 00   | 31   | 0A   | 0A   | 00   | 08   | 00   | 00       | 00           |
| 00   | 00   | 00   | 33   | 1B   | 1B   | 00   | 08   | 00   | 00   | 00       | 2F           |
| 00   | 00   | 35   | 0C   | 0C   | 00   | 1C   | 00   | 00   | 00   | 34       | 0B           |
| 00   | 37   | 1D   | 1D   | 00   | 0D   | 00   | 00   | 00   | 36   | 0D       | 0D           |
|      |      |      |      |      |      |      |      |      |      | 00       | 1D           |
|      |      |      |      |      |      |      |      |      |      | 00       | 00           |
| 0A   | 00   | 00   | 00   | 23   | 0A   | 0A   | 00   | 08   | 00   | 00       | 00           |
| 00   | 00   | 00   | 25   | 1B   | 1B   | 00   | 08   | 00   | 00   | 00       | 21           |
| 00   | 00   | 27   | 0C   | 0C   | 00   | 1C   | 00   | 00   | 00   | 26       | 0B           |
| 00   | 29   | 1D   | 1D   | 00   | 0D   | 00   | 00   | 00   | 28   | 0D       | 0D           |
|      |      |      |      |      |      |      |      |      |      | 00       | 1D           |
|      |      |      |      |      |      |      |      |      |      | 00       | 00           |
| 07   | 00   | 00   | 00   | BE   | 07   | 07   | 00   | 06   | 00   | 00       | 00           |
| 00   | 00   | 01   | 0A   | 09   | 09   | 00   | 15   | 00   | 00   | 00       | BD           |
| 00   | 00   | BF   | 08   | 08   | 00   | 1A   | 00   | 00   | 01   | 0C       | 15           |
| 00   | C1   | 0B   | 0B   | 00   | 0A   | 00   | 00   | 00   | C0   | 0A       | 1A           |
| C3   | 1C   | 1C   | 00   | 1B   | 00   | 00   | 00   | C2   | 1B   | 1B       | 00           |
| 0D   | 0D   | 00   | 0C   | 00   | 00   | 00   | C4   | 0C   | 0C   | 00       | 0B           |
|      |      |      |      |      |      |      |      |      |      | 1C       | 00           |
|      |      |      |      |      |      |      |      |      |      | 00       | 00           |
|      |      |      |      |      |      |      |      |      |      | 00       | 00           |
|      |      |      |      |      |      |      |      |      |      | 00       | C5           |
| 07   | 00   | 00   | 00   | B2   | 07   | 07   | 00   | 06   | 00   | 00       | 00           |
| 00   | 00   | 01   | 0B   | 09   | 09   | 00   | 15   | 00   | 00   | 00       | B1           |
| 00   | 00   | B3   | 0B   | 0B   | 00   | 1A   | 00   | 00   | 01   | 0D       | 15           |
| 00   | B6   | 0B   | 0B   | 00   | 0A   | 00   | 00   | 00   | B5   | 0A       | 1A           |
| B8   | 1C   | 1C   | 00   | 1B   | 00   | 00   | 00   | B7   | 1B   | 1B       | 00           |
| 0D   | 0D   | 00   | 0C   | 00   | 00   | 00   | B9   | 0C   | 0C   | 00       | 0B           |
|      |      |      |      |      |      |      |      |      |      | 1C       | 00           |
|      |      |      |      |      |      |      |      |      |      | 00       | 00           |
|      |      |      |      |      |      |      |      |      |      | 00       | 00           |
|      |      |      |      |      |      |      |      |      |      | 00       | BA           |

|       |       |       |       |       |      |      |      |     |
|-------|-------|-------|-------|-------|------|------|------|-----|
| 10    | 2088  | 2107  | 2097  | 2095  | 770  | 1027 | 2052 | -   |
| 1798  | 2055  | 2568  | 2826  | 6922  | 3082 | 7174 | -    |     |
| 3339  | 7195  | 7451  | 3340  | 7436  | 0    |      |      |     |
| 10    | 1538  | 774   | 1795  | 1031  | 2052 | 2568 | 2826 | -   |
| 6922  | 7174  | 7195  | 3100  | 3340  | 7436 | -    |      |     |
| 12045 | 12061 | 12591 | 15153 | 10299 | 552  | 0    |      |     |
| 3     | 6923  | 7435  | 3355  | 7437  | 0    |      |      |     |
| 16    | 8     | 8     | 5     | 0     | 0    | 0    | 8    | 0   |
| 0     | 10    | 0     | 11    | 11    | 8    | 0    | 0    | 0   |
| 27    | 9     | 0     | 0     | 0     | 27   | 0    | 28   | 28  |
| 0     | 28    | 0     | 12    | 12    | 11   | 0    | 0    | 12  |
| 13    | 12    | 0     | 0     | 0     | 13   | 0    | 29   | 29  |
| 0     | 29    | 0     |       |       |      |      |      |     |
| 16    | 8     | 8     | 19    | 0     | 0    | 0    | 8    | 0   |
| 0     | 0     | 10    | 0     | 11    | 11   | 22   | 0    | 0   |
| 27    | 27    | 23    | 0     | 0     | 0    | 27   | 0    | 28  |
| 0     | 0     | 0     | 28    | 0     | 12   | 12   | 25   | 0   |
| 0     | 13    | 13    | 26    | 0     | 0    | 0    | 13   | 0   |
| 27    | 0     | 0     | 0     | 29    | 0    |      |      |     |
| 16    | 8     | 8     | 47    | 0     | 0    | 0    | 8    | 0   |
| 0     | 0     | 10    | 0     | 11    | 11   | 50   | 0    | 0   |
| 27    | 27    | 51    | 0     | 0     | 0    | 27   | 0    | 28  |
| 0     | 0     | 0     | 28    | 0     | 12   | 12   | 53   | 0   |
| 0     | 13    | 13    | 54    | 0     | 0    | 0    | 13   | 0   |
| 55    | 0     | 0     | 0     | 29    | 0    |      |      |     |
| 16    | 8     | 8     | 33    | 0     | 0    | 0    | 8    | 0   |
| 0     | 0     | 10    | 0     | 11    | 11   | 36   | 0    | 0   |
| 27    | 27    | 37    | 0     | 0     | 0    | 27   | 0    | 28  |
| 0     | 0     | 0     | 28    | 0     | 12   | 12   | 39   | 0   |
| 0     | 13    | 13    | 40    | 0     | 0    | 0    | 13   | 0   |
| 41    | 0     | 0     | 0     | 29    | 0    |      |      |     |
| 26    | 6     | 6     | -67   | 0     | 0    | 0    | 6    | 0   |
| 0     | 0     | 7     | 0     | 21    | 21   | -10  | 0    | 0   |
| 9     | 9     | 10    | 1     | 0     | 0    | 9    | 0    | 26  |
| 0     | 0     | 26    | 0     | 8     | 8    | -65  | 0    | 0   |
| 10    | 10    | -64   | 0     | 0     | 0    | 10   | 0    | 11  |
| 0     | 0     | 0     | 11    | 0     | 27   | 27   | -62  | 0   |
| 0     | 28    | 28    | -61   | 0     | 0    | 0    | 28   | 0   |
| -60   | 0     | 0     | 0     | 12    | 0    | 13   | 13   | -59 |
| 0     | 13    | 0     | 29    | 29    | -58  | 0    | 0    | 0   |
| 26    | 6     | 6     | -79   | 0     | 0    | 0    | 6    | 0   |
| 0     | 0     | 7     | 0     | 21    | 21   | -9   | 0    | 0   |
| 9     | 9     | 11    | 1     | 0     | 0    | 9    | 0    | 26  |
| 0     | 0     | 26    | 0     | 8     | 8    | -77  | 0    | 0   |
| 10    | 10    | -75   | 0     | 0     | 0    | 10   | 0    | 11  |
| 0     | 0     | 0     | 11    | 0     | 27   | 27   | -73  | 0   |
| 0     | 28    | 28    | -72   | 0     | 0    | 0    | 28   | 0   |

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| 08 | 00 | 00 | 00 | 41 | 08 | 08 | 00 | 07 | 00 | 00 | 00 | 40 | 07 | 07 | 053F0 |
| 00 | 00 | 00 | 43 | 08 | 08 | 00 | 0A | 00 | 00 | 00 | 42 | 0A | 08 | 00 | 053F4 |
| 00 | 00 | 45 | 1C | 08 | 00 | 1B | 00 | 00 | 00 | 44 | 1B | 08 | 00 | 08 | 05403 |
| 00 | 47 | 0D | 08 | 00 | 0C | 00 | 00 | 00 | 46 | 0C | 08 | 00 | 1C | 00 | 05412 |
| 49 | 0A | 0A | 00 | 1D | 00 | 00 | 00 | 48 | 1D | 08 | 00 | 0D | 00 | 00 | 05421 |
| 0A | 0B | 00 | 0B | 00 | 00 | 4A | 0B | 0A | 00 | 0A | 00 | 00 | 00 | 00 | 05430 |
| 1B | 00 | 0B | 00 | 00 | 00 | 4C | 0B | 0B | 00 | 0B | 00 | 00 | 00 | 4B | 0543F |
| 00 | 1C | 00 | 00 | 00 | 4E | 1C | 1C | 00 | 1B | 00 | 00 | 00 | 4D | 1B | 0544E |
| 0D | 00 | 00 | 00 | 50 | 0D | 0D | 00 | 0C | 00 | 00 | 00 | 4F | 0C | 0C | 0545D |
|    |    |    |    |    |    |    | 1D | 00 | 00 | 00 | 51 | 1D | 1D | 00 | 0546C |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0547B |

P.AJV: .LONG  
.BYTE

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| 28 | 00 | 00 | 00 | 61 | 08 | 08 | 00 | 28 | 00 | 00 | 00 | 54 | 0E | 0E | 05484 |
| 00 | 00 | 00 | 63 | 0B | 0B | 00 | 28 | 00 | 00 | 00 | 62 | 0A | 0A | 00 | 05488 |
| 00 | 00 | 65 | 1C | 1C | 00 | 28 | 00 | 00 | 00 | 64 | 1B | 1B | 00 | 28 | 05497 |
| 00 | 67 | 0D | 0D | 00 | 28 | 00 | 00 | 00 | 66 | 0C | 0C | 00 | 28 | 00 | 054A6 |
|    |    |    |    | 00 | 28 | 00 | 00 | 68 | 1D | 1D | 00 | 28 | 00 | 00 | 054B5 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 054C4 |

P.AJW: .LONG  
.BYTE

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| 28 | 00 | 00 | 00 | 6C | 08 | 08 | 00 | 28 | 00 | 00 | 00 | 5E | 0E | 0E | 054D0 |
| 00 | 00 | 00 | 6E | 0B | 0B | 00 | 28 | 00 | 00 | 00 | 6D | 0A | 0A | 00 | 054D4 |
| 00 | 00 | 70 | 1C | 1C | 00 | 28 | 00 | 00 | 00 | 6F | 1B | 1B | 00 | 28 | 054E3 |
| 00 | 72 | 0D | 0D | 00 | 28 | 00 | 00 | 00 | 71 | 0C | 0C | 00 | 28 | 00 | 054F2 |
|    |    |    |    | 00 | 28 | 00 | 00 | 73 | 1D | 1D | 00 | 28 | 00 | 00 | 05501 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 05510 |

P.AJX: .LONG  
.BYTE

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| 28 | 00 | 00 | 00 | 8F | 08 | 08 | 00 | 28 | 00 | 00 | 00 | 5C | 0E | 0E | 0551C |
| 00 | 00 | 00 | 92 | 0B | 0B | 00 | 28 | 00 | 00 | 00 | 91 | 0A | 0A | 00 | 05520 |
| 00 | 00 | 94 | 1C | 1C | 00 | 28 | 00 | 00 | 00 | 93 | 1B | 1B | 00 | 28 | 0552F |
|    |    |    |    |    |    |    |    |    |    |    |    | 00 | 28 | 00 | 0553E |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0554D |

P.AJY: .LONG  
.BYTE

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| 28 | 00 | 00 | 00 | 7F | 08 | 08 | 00 | 28 | 00 | 00 | 00 | 58 | 0E | 0E | 05550 |
| 00 | 00 | 00 | 82 | 0B | 0B | 00 | 28 | 00 | 00 | 00 | 81 | 0A | 0A | 00 | 05554 |
| 00 | 00 | 84 | 1C | 1C | 00 | 28 | 00 | 00 | 00 | 83 | 1B | 1B | 00 | 28 | 05563 |
|    |    |    |    |    |    |    |    |    |    |    |    | 00 | 28 | 00 | 05572 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 05581 |

P.AJZ: .LONG  
.BYTE

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| 28 | 00 | 00 | 00 | 87 | 08 | 08 | 00 | 28 | 00 | 00 | 00 | 5A | 0E | 0E | 05584 |
| 00 | 00 | 00 | 8A | 0B | 0B | 00 | 28 | 00 | 00 | 00 | 89 | 0A | 0A | 00 | 05588 |
| 00 | 00 | 8C | 1C | 1C | 00 | 28 | 00 | 00 | 00 | 8B | 1B | 1B | 00 | 28 | 05597 |
|    |    |    |    |    |    |    |    |    |    |    |    | 00 | 28 | 00 | 055A6 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 055B5 |

P.AKA: .LONG  
.BYTE

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| 28 | 00 | 00 | 00 | 77 | 08 | 08 | 00 | 28 | 00 | 00 | 00 | 56 | 0E | 0E | 055B8 |
| 00 | 00 | 00 | 7A | 0B | 0B | 00 | 28 | 00 | 00 | 00 | 79 | 0A | 0A | 00 | 055BC |
| 00 | 00 | 7C | 1C | 1C | 00 | 28 | 00 | 00 | 00 | 7B | 1B | 1B | 00 | 28 | 055CB |
|    |    |    |    |    |    |    |    |    |    |    |    | 00 | 28 | 00 | 055DA |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 055E9 |

P.AKB: .LONG  
.BYTE

|    |    |    |    |    |    |    |    |                    |
|----|----|----|----|----|----|----|----|--------------------|
| 00 | 08 | 00 | 00 | 00 | 97 | 08 | 08 | 055EC              |
|    |    |    |    |    |    |    |    | 055F0 P.AKC: .LONG |
|    |    |    |    |    |    |    |    | .BYTE              |

```

-71 0. 0. 0. 12. 0. 13. 13. -70. 0. 0. -
0. 13. 0. 29. 29. -69. 0. 0. 0. 29. 0. -
36
7. 7. 64. 0. 0. 0. 7. 0. 8. 8. 65. 0. 0. -
0. 8. 0. 8. 10. 66. 0. 0. 0. 10. 0. 8. 0. -
11. 67. 0. 0. 0. 11. 0. 8. 27. 68. 0. 0. -
0. 27. 0. 8. 28. 69. 0. 0. 0. 28. 0. 8. -
12. 70. 0. 0. 0. 12. 0. 8. 13. 71. 0. 0. -
0. 13. 0. 8. 29. 72. 0. 0. 0. 29. 0. 10. -
10. 73. 0. 0. 0. 10. 0. 10. 11. 74. 0. 0. -
0. 11. 0. 11. 10. 75. 0. 0. 0. 11. 0. 11. -
11. 76. 0. 0. 0. 11. 0. 27. 27. 77. 0. 0. -
0. 27. 0. 28. 28. 78. 0. 0. 0. 28. 0. 12. -
12. 79. 0. 0. 0. 12. 0. 13. 13. 80. 0. 0. -
0. 13. 0. 29. 29. 81. 0. 0. 0. 29. 0. -
18
14. 14. 84. 0. 0. 0. 40. 0. 8. 8. 97. 0. -
0. 0. 40. 0. 10. 10. 98. 0. 0. 0. 40. 0. -
11. 11. 99. 0. 0. 0. 40. 0. 27. 27. 100. -
0. 0. 0. 40. 0. 28. 28. 101. 0. 0. 0. 40. -
0. 12. 12. 102. 0. 0. 0. 40. 0. 13. 13. -
103. 0. 0. 0. 40. 0. 29. 29. 104. 0. 0. -
0. 40. 0. -
18
14. 14. 94. 0. 0. 0. 40. 0. 8. 8. 108. 0. -
0. 0. 40. 0. 10. 10. 109. 0. 0. 0. 40. 0. -
11. 11. 110. 0. 0. 0. 40. 0. 27. 27. 111. -
0. 0. 0. 40. 0. 28. 28. 112. 0. 0. 0. 40. -
0. 12. 12. 113. 0. 0. 0. 40. 0. 13. 13. -
114. 0. 0. 0. 40. 0. 29. 29. 115. 0. 0. -
0. 40. 0. -
12
14. 14. 92. 0. 0. 0. 40. 0. 8. 8. -113. -
0. 0. 0. 40. 0. 10. 10. -111. 0. 0. 0. -
40. 0. 11. 11. -110. 0. 0. 0. 40. 0. 27. -
27. -109. 0. 0. 0. 40. 0. 28. 28. -108. -
0. 0. 0. 40. 0. -
12
14. 14. 88. 0. 0. 0. 40. 0. 8. 8. 127. 0. -
0. 0. 40. 0. 10. 10. -127. 0. 0. 0. 40. -
0. 11. 11. -126. 0. 0. 0. 40. 0. 27. 27. -
-125. 0. 0. 0. 40. 0. 28. 28. -124. 0. 0. -
0. 40. 0. -
12
14. 14. 90. 0. 0. 0. 40. 0. 8. 8. -121. -
0. 0. 0. 40. 0. 10. 10. -119. 0. 0. 0. -
40. 0. 11. 11. -118. 0. 0. 0. 40. 0. 27. -
27. -117. 0. 0. 0. 40. 0. 28. 28. -116. -
0. 0. 0. 40. 0. -
12
14. 14. 86. 0. 0. 0. 40. 0. 8. 8. 119. 0. -
0. 0. 40. 0. 10. 10. 121. 0. 0. 0. 40. -
11. 11. 122. 0. 0. 0. 40. 0. 27. 27. 123. -
0. 0. 0. 40. 0. 28. 28. 124. 0. 0. 0. 40. -
0. -
2
8. 8. -105. 0. 0. 0. 8. 0

```



```

000051FC 000051A4 000055FC 057C4 .LONG 22012, 20900, 20988
                                01 057D0 .BYTE 1
                                00# 057D1 .BYTE 0[3]
000051FC 000051A4 00005608 057D4 .LONG 22024, 20900, 20988
                                01 057E0 .BYTE 1
                                00# 057E1 .BYTE 0[3]
000051FC 000051A4 00005614 057E4 .LONG 22036, 20900, 20988
                                01 057F0 .BYTE 1
                                00# 057F1 .BYTE 0[115]
000051FC 000051A4 0000562C 05864 .LONG 22060, 20900, 20988
                                01 05870 .BYTE 1
                                00# 05871 .BYTE 0[227]
000051FC 000051D0 00000000 05954 .LONG 0, 20944, 20988
                                01 05960 .BYTE 1
                                00# 05961 .BYTE 0[3]
000051FC 000051A4 00000000 05964 .LONG 0, 20900, 20988
                                01 05970 .BYTE 1
                                00# 05971 .BYTE 0[67]
00000000 00000000 00000000 05984 .LONG 0, 0, 0
                                01 059C0 .BYTE 1
                                059C1 .BLKB 179

```

.PSECT DBG\$OWN,NOEXE, PIC,2

```

00000000 00000 BLISS_BITSELECTION_FLAG1:
                                .LONG 0
00000000 00004 BLISS_BITSELECTION_FLAG2:
                                .LONG 0
00008 BLISS_INDIRECTION_FLAG:
                                .BLKB 4
0000C CVT_ROUND_FLAG:
                                .BLKB 4
00010 CVT_TBL: .BLKB 4
00014 CVT_TBL_SIZE:
                                .BLKB 4
00018 CVTINFO_TABLE:
                                .BLKB 4
0001C MAP_TBL: .BLKB 4
00020 MAP_TBL_SIZE:
                                .BLKB 4
00024 MAX_DEPTH:
                                .BLKB 4
00028 OPINFO_TABLE:
                                .BLKB 4
0002C SAVE_RESULT_DESC:
                                .BLKB 12

```

.PSECT DBG\$GLOBAL,NOEXE, PIC,2

```

00000 DBG$GL_OPCODE_NAME::
                                .BLKB 4

```

```

TABLEBASE= P.AAA
ADA_HIER_TABLE= P.AAB
ADA_DEPOSIT_TABLE= P.AAC
ADA_UNARY_PCUS_TABLE= P.AAD

```

```

ADA_UNARY_MINUS_TABLE=
ADA_ABSOLUTE_TABLE= P.AAE
ADA_ADD_TABLE= P.AAF
ADA_SUBTRACT_TABLE= P.AAG
ADA_MULTIPLY_TABLE= P.AAH
ADA_DIVIDE_TABLE= P.AAI
ADA_MODULUS_TABLE= P.AAJ
ADA_REMAINDER_TABLE= P.AAK
ADA_POWER_OF_TABLE= P.AAL
ADA_NOT_TABLE= P.AAM
ADA_AND_TABLE= P.AAN
ADA_OR_TABLE= P.AAO
ADA_XOR_TABLE= P.AAP
ADA_EQUAL_TABLE= P.AAQ
ADA_NOT_EQUAL_TABLE= P.AAR
ADA_LSS_THAN_TABLE= P.AAS
ADA_GTR_THAN_TABLE= P.AAT
ADA_LSS_EQUAL_TABLE= P.AAU
ADA_GTR_EQUAL_TABLE= P.AAV
ADA_CONCATENATE_TABLE= P.AAW
BASIC_HIER1_TABLE= P.AAX
BASIC_HIER2_TABLE= P.AAY
BASIC_HIERD_TABLE= P.AAZ
BASIC_ADD_TABLE= P.ABA
BASIC_SUB_TABLE= P.ABB
BASIC_MUL_TABLE= P.ABC
BASIC_DIV_TABLE= P.ABD
BASIC_UNARY_PLUS_TABLE= P.ABE
BASIC_UNARY_MINUS_TABLE= P.ABF
BASIC_POWER_TABLE= P.ABG
BASIC_EQL_TABLE= P.ABH
BASIC_NEQ_TABLE= P.ABI
BASIC_GTR_TABLE= P.ABJ
BASIC_GEQ_TABLE= P.ABK
BASIC_LSS_TABLE= P.ABL
BASIC_LEQ_TABLE= P.ABM
BASIC_BIT_NOT_TABLE= P.ABN
BASIC_BIT_AND_TABLE= P.ABO
BASIC_BIT_OR_TABLE= P.ABP
BASIC_BIT_XOR_TABLE= P.ABQ
BASIC_BIT_EQV_TABLE= P.ABR
BASIC_BIT_IMP_TABLE= P.ABS
BLISS_MAP_TABLE= P.ABT
BLISS_HIER_TABLE= P.ABU
BLISS_HIERD_TABLE= P.ABV
BLISS_ADD_TABLE= P.ABW
BLISS_SUB_TABLE= P.ABX
BLISS_MUL_TABLE= P.ABY
BLISS_DIV_TABLE= P.ABZ
BLISS_MOD_TABLE= P.ACA
BLISS_SHIFT_TABLE= P.ACB
BLISS_EQUAL_TABLE= P.ACC
BLISS_NOT_EQUAL_TABLE= P.ACD

```

```

BLISS_LSS_THAN_TABLE= P.ACE
BLISS_LSSU_THAN_TABLE= P.ACF
BLISS_GTR_THAN_TABLE= P.ACG
BLISS_GTRU_THAN_TABLE= P.ACH
BLISS_LSS_EQUAL_TABLE= P.ACI
BLISS_LSSU_EQUAL_TABLE= P.ACJ
BLISS_GTR_EQUAL_TABLE= P.ACK
BLISS_GTRU_EQUAL_TABLE= P.ACL
BLISS_BIT_AND_TABLE= P.ACM
BLISS_BIT_OR_TABLE= P.ACO
BLISS_BIT_XOR_TABLE= P.ACP
BLISS_BIT_EQV_TABLE= P.ACQ
BLISS_UNARY_PLUS_TABLE= P.ACR
BLISS_UNARY_MINUS_TABLE= P.ACS
BLISS_BIT_NOT_TABLE= P.ACT
BLISS_BITSELECT_TABLE= P.ACU
BLISS_INDIRECT_TABLE= P.ACV
C_HIER_TABLE= P.ACW
C_HIERD_TABLE= P.ACX
C_ADD_TABLE= P.ACY
C_SUB_TABLE= P.ACZ
C_MUL_TABLE= P.ADA
C_DIV_TABLE= P.ADB
C_UNARY_MINUS_TABLE= P.ADC
C_UNARY_PLUS_TABLE= P.ADD
C_MOD_TABLE= P.ADE
C_EQL_TABLE= P.ADF
C_NEQ_TABLE= P.ADG
C_LSS_TABLE= P.ADH
C_GTR_TABLE= P.ADI
C_LEQ_TABLE= P.ADJ
C_GEQ_TABLE= P.ADK
C_BIT_AND_TABLE= P.ADL
C_BIT_OR_TABLE= P.ADM
C_BIT_XOR_TABLE= P.ADN
C_BIT_NOT_TABLE= P.ADO
C_AND_TABLE= P.ADP
C_OR_TABLE= P.ADQ
C_NOT_TABLE= P.ADR
C_SHIFT_LEFT_TABLE= P.ADS
C_SHIFT_RT_TABLE= P.ADT
C_PRE_INCR_TABLE= P.ADU
C_POST_INCR_TABLE= P.ADV
C_PRE_DECR_TABLE= P.ADW

```

C\_POST DECR TABLE= P.ADX  
 C\_ADDRESS TABLE= P.ADY  
 C\_SIZEOF TABLE= P.ADZ  
 C\_INDIRECT TABLE= P.AEA  
 COBOL\_CVT TABLE= P.AEB  
 COBOL\_HIER TABLE= P.AEC  
 COBOL\_HIERD TABLE= P.AED  
 COBOL\_ADD TABLE= P.AEE  
 COBOL\_SUB TABLE= P.AEF  
 COBOL\_MUL TABLE= P.AEG  
 COBOL\_DIV TABLE= P.AEH  
 COBOL\_UNARY\_PLUS\_TABLE= P.AEI  
 COBOL\_UNARY\_MINUS\_TABLE= P.AEJ  
 COBOL\_EQL TABLE= P.AEK  
 COBOL\_NEQ TABLE= P.AEL  
 COBOL\_GTR TABLE= P.AEM  
 COBOL\_GEQ TABLE= P.AEN  
 COBOL\_LSS TABLE= P.AEO  
 COBOL\_LEQ TABLE= P.AEP  
 COBOL\_NOT TABLE= P.AEQ  
 COBOL\_AND TABLE= P.AER  
 COBOL\_OR TABLE= P.AES  
 FORTRAN\_MAP TABLE= P.AET  
 FORTRAN\_HIER1 TABLE= P.AEU  
 FORTRAN\_HIER2 TABLE= P.AEV  
 FORTRAN\_HIER3 TABLE= P.AEW  
 FORTRAN\_HIERD TABLE= P.AEX  
 FORTRAN\_INCOMP\_TABLE= P.AEY  
 FORTRAN\_ADD TABLE= P.AEZ  
 FORTRAN\_SUB TABLE= P.AFA  
 FORTRAN\_MUL TABLE= P.AFB  
 FORTRAN\_DIV TABLE= P.AFC  
 FORTRAN\_UNARY\_PLUS\_TABLE= P.AFD  
 FORTRAN\_UNARY\_MINUS\_TABLE= P.AFE  
 FORTRAN\_POWER TABLE= P.AFF  
 FORTRAN\_CONCAT\_TABLE= P.AFG  
 FORTRAN\_EQL TABLE= P.AFH  
 FORTRAN\_NEQ TABLE= P.AFI  
 FORTRAN\_GTR TABLE= P.AFJ  
 FORTRAN\_GEQ TABLE= P.AFK  
 FORTRAN\_LSS TABLE= P.AFL  
 FORTRAN\_LEQ TABLE= P.AFM  
 FORTRAN\_BIT\_NOT\_TABLE= P.AFN  
 FORTRAN\_BIT\_AND\_TABLE= P.AFO  
 FORTRAN\_BIT\_OR\_TABLE= P.AFP  
 FORTRAN\_BIT\_XOR\_TABLE= P.AFQ  
 FORTRAN\_BIT\_EQV\_TABLE=

```

MACRO_MAP_TABLE= P.AFR
MACRO_HIER_TABLE= P.AFS
MACRO_HIERD_TABLE= P.AFT
MACRO_ADD_TABLE= P.AFU
MACRO_SUB_TABLE= P.AFV
MACRO_MUL_TABLE= P.AFW
MACRO_DIV_TABLE= P.AFX
MACRO_SHIFT_TABLE= P.AFY
MACRO_MOD_TABLE= P.AFZ
MACRO_UNARY_PLUS_TABLE= P.AGA
MACRO_UNARY_MINUS_TABLE= P.AGB
MACRO_EQUAL_TABLE= P.AGC
MACRO_NOT_EQUAL_TABLE= P.AGD
MACRO_LSS_THAN_TABLE= P.AGE
MACRO_GTR_THAN_TABLE= P.AGF
MACRO_LSS_EQUAL_TABLE= P.AGG
MACRO_GTR_EQUAL_TABLE= P.AGH
MACRO_LSSU_THAN_TABLE= P.AGI
MACRO_GTRU_THAN_TABLE= P.AGJ
MACRO_LSSU_EQUAL_TABLE= P.AGK
MACRO_GTRU_EQUAL_TABLE= P.AGL
MACRO_BIT_AND_TABLE= P.AGM
MACRO_BIT_OR_TABLE= P.AGN
MACRO_BIT_XOR_TABLE= P.AGO
MACRO_BIT_EQV_TABLE= P.AGP
MACRO_BIT_NOT_TABLE= P.AGQ
MACRO_BITSELECT_TABLE= P.AGR
MACRO_INDIRECT_TABLE= P.AGS
PASCAL_MAP_TABLE= P.AGT
PASCAL_HIER_TABLE= P.AGU
PASCAL_HIERT_TABLE= P.AGV
PASCAL_HIERD_TABLE= P.AGW
PASCAL_INCOMP_TABLE= P.AGX
PASCAL_ADD_TABLE= P.AGY
PASCAL_SUB_TABLE= P.AGZ
PASCAL_MUL_TABLE= P.AHA
PASCAL_POWER_TABLE= P.AHB
PASCAL_DIV_TABLE= P.AHC
PASCAL_UNARY_PLUS_TABLE= P.AHD
PASCAL_UNARY_MINUS_TABLE= P.AHE
PASCAL_INTDIV_TABLE= P.AHF

```

```

PASCAL_MOD_TABLE= P.AHH
PASCAL_REM_TABLE= P.AHI
PASCAL_EQL_TABLE= P.AHJ
PASCAL_NEQ_TABLE= P.AHK
PASCAL_GEQ_TABLE= P.AHL
PASCAL_GTR_TABLE= P.AHM
PASCAL_LSS_TABLE= P.AHN
PASCAL_LEQ_TABLE= P.AHO
PASCAL_NOT_TABLE= P.AHP
PASCAL_AND_TABLE= P.AHQ
PASCAL_OR_TABLE= P.AHR
PASCAL_IN_TABLE= P.AHS
PASCAL_SUCCESOR_TABLE=
P.AHT
PASCAL_PREDECESSOR_TABLE=
P.AHU
PLI_MAP_TABLE= P.AHV
PLI_INCOMP_TABLE= P.AHW
PLI_CVT_TABLE= P.AHX
PLI_HIER1_TABLE= P.AHY
PLI_HIER2_TABLE= P.AHZ
PLI_HIER3_TABLE= P.AIA
PLI_HIER4_TABLE= P.AIB
PLI_HIER5_TABLE= P.AIC
PLI_HIERD_TABLE= P.AID
PLI_ADD_TABLE= P.AIE
PLI_SUB_TABLE= P.AIF
PLI_MUL_TABLE= P.AIG
PLI_DIV_TABLE= P.AIH
PLI_UNARY_PLUS_TABLE=
P.AII
PLI_UNARY_MINUS_TABLE=
P.AIJ
PLI_POWER_TABLE= P.AIK
PLI_CONCAT_TABLE= P.AIL
PLI_EQL_TABLE= P.AIM
PLI_NEQ_TABLE= P.AIN
PLI_GTR_TABLE= P.AIO
PLI_GEQ_TABLE= P.AIP
PLI_LSS_TABLE= P.AIQ
PLI_LEQ_TABLE= P.AIR
PLI_BIT_NOT_TABLE= P.AIS
PLI_BIT_AND_TABLE= P.AIT
PLI_BIT_OR_TABLE= P.AIU
RPG_HIER_TABLE= P.AIV
RPG_HIERD_TABLE= P.AIW
RPG_ADD_TABLE= P.AIX
RPG_SUB_TABLE= P.AIY
RPG_MUL_TABLE= P.AIZ
RPG_DIV_TABLE= P.AJA
RPG_UNARY_PLUS_TABLE=
P.AJB
RPG_UNARY_MINUS_TABLE=
P.AJC
RPG_EQL_TABLE= P.AJD
RPG_NEQ_TABLE= P.AJE
RPG_GTR_TABLE= P.AJF

```

K 6  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 B11ss-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32;1

RPG\_GEQ\_TABLE= P.AJG  
RPG\_LSS\_TABLE= P.AJH  
RPG\_LEQ\_TABLE= P.AJI  
RPG\_NOT\_TABLE= P.AJJ  
RPG\_AND\_TABLE= P.AJK  
RPG\_OR\_TABLE= P.AJL  
UNKNOWN\_HIER\_TABLE= P.AJM  
UNKNOWN\_HIERD\_TABLE= P.AJN  
UNKNOWN\_INCOMP\_TABLE=  
P.AJO  
UNKNOWN\_ADD\_TABLE= P.AJP  
UNKNOWN\_SUB\_TABLE= P.AJQ  
UNKNOWN\_MUL\_TABLE= P.AJR  
UNKNOWN\_DIV\_TABLE= P.AJS  
UNKNOWN\_UNARY\_PLUS\_TABLE=  
P.AJT  
UNKNOWN\_UNARY\_MINUS\_TABLE=  
P.AJU  
UNKNOWN\_POWER\_TABLE= P.AJV  
UNKNOWN\_EQL\_TABLE= P.AJW  
UNKNOWN\_NEQ\_TABLE= P.AJX  
UNKNOWN\_LSS\_THAN\_TABLE=  
P.AJY  
UNKNOWN\_GTR\_THAN\_TABLE=  
P.AJZ  
UNKNOWN\_LEQ\_TABLE= P.AKA  
UNKNOWN\_GEQ\_TABLE= P.AKB  
UNKNOWN\_AND\_TABLE= P.AKC  
UNKNOWN\_OR\_TABLE= P.AKD  
UNKNOWN\_XOR\_TABLE= P.AKE  
UNKNOWN\_EQV\_TABLE= P.AKF  
UNKNOWN\_NOT\_TABLE= P.AKG  
UNKNOWN\_CONCATENATE\_TABLE=  
P.AKH

.EXTRN FORSCVT\_D\_TE, FORSCVT\_G\_TE  
.EXTRN FORSCVT\_H\_TE, MTHSJNOT  
.EXTRN OTSSCVT\_TB\_L, OTSSCVT\_TI\_L  
.EXTRN OTSSCVT\_T\_F, OTSSCVT\_T\_D  
.EXTRN OTSSCVT\_T\_G, OTSSCVT\_T\_H  
.EXTRN OTSSCVT\_TD\_L, OTSSCVT\_TZ\_L  
.EXTRN PLISCHARABIT\_R6  
.EXTRN PLISCVT\_ANY, DBGSCollect  
.EXTRN DBGSCOVER\_DX\_DX  
.EXTRN DBGSCVT\_DX\_DX, DBGSCVT\_TQUADWORD\_TO\_VALUE  
.EXTRN DBGSCVT\_TQUADWORD\_TO\_VALUE  
.EXTRN DBGSCVT\_TOCTAWORD\_TO\_VALUE  
.EXTRN DBGSCVT\_TRFA\_TO\_VALUE  
.EXTRN DBGSGET\_DST\_NAME  
.EXTRN DBGSGET\_TEMPMEM  
.EXTRN DBGSMKE\_SKELETON\_DESC  
.EXTRN DBGSMKE\_VAL\_DESC  
.EXTRN DBGSMKE\_VMS\_DESC  
.EXTRN DBGSNEWLINE, DBGSPERFORM\_OPERATOR  
.EXTRN DBGSPRIM\_TO\_ADDR  
.EXTRN DBGSPRIM\_TO\_VAL  
.EXTRN DBGSPRINT, DBGSSYA\_SYMVALUE  
.EXTRN DBGSSYA\_TYP\_ATOMIC

DBGEVALOP  
V04-000

L 6  
16-Sep-1984 00:32:25 VAX-11 B11ss-32 V4.0-742  
5-Sep-1984 21:54:24 [DEBUG.SRC]DBGEVALOP.B32;1

Page 151  
(22)

.EXTRN DBG\$STA\_TYP\_ENUM  
.EXTRN DBG\$STA\_TYP\_PICT  
.EXTRN DBG\$STA\_TYP\_SET  
.EXTRN DBG\$STA\_TYP\_SUBRNG  
.EXTRN DBG\$STA\_TYP\_TYPEDPTR  
.EXTRN DBG\$TYPEID\_FOR\_SET  
.EXTRN DBG\$GB\_LANGUAGE  
.EXTRN DBG\$GL\_NEG\_CONST\_TOKEN  
.EXTRN DBG\$GL\_POS\_CONST\_TOKEN  
.EXTRN DBG\$GL\_NEG\_SIGN\_TOKEN  
.EXTRN DBG\$GL\_POS\_SIGN\_TOKEN  
.EXTRN DBG\$GL\_DEVELOPER

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

0000 00000 AAA\_DUMMY:

50 D4 00002  
04 00004

.WORD Save nothing  
CLRL R0  
RET

: 5709  
: 5717  
:

; Routine Size: 5 bytes, Routine Base: DBG\$CODE + 0000

```
5607 5718 1 GLOBAL ROUTINE DBG$BLISS_BITSELECT (OPERATOR, ARG_DESC, RESULT_DESC): NOVALUE =
5608 5719 1
5609 5720 1 FUNCTION
5610 5721 1
5611 5722 1     This routine is called from DBG$PERFORM_OPERATOR to do the
5612 5723 1     BLISS bit select operation (e.g. X<p,s,e>). It receives
5613 5724 1     the operator token containing the p,s,e information and
5614 5725 1     a pointer to the descriptor for X. It builds a new descriptor with
5615 5726 1     the new bit offset, length, and sign extension information.
5616 5727 1
5617 5728 1 INPUTS
5618 5729 1
5619 5730 1     OPERATOR      - points to the operator token for the BITSELECT operator.
5620 5731 1                   The p,s,e information can be extracted from the token
5621 5732 1                   in the BIT_OFFSET, BIT_LENGTH, and SGNEXT fields.
5622 5733 1     ARG_DESC      - points to the VMS descriptor representing the argument
5623 5734 1                   of the bit-select operator.
5624 5735 1     RESULT_DESC   - points to the VMS descriptor representing the result.
5625 5736 1
5626 5737 1 OUTPUTS
5627 5738 1
5628 5739 1     The result VMS descriptor is filled in.
5629 5740 1     No value is returned.
5630 5741 1
5631 5742 2 BEGIN
5632 5743 2 MAP
5633 5744 2     OPERATOR      : REF TOKEN$ENTRY,
5634 5745 2     ARG_DESC      : REF DBG$STG_DESC,
5635 5746 2     RESULT_DESC   : REF DBG$STG_DESC;
5636 5747 2
5637 5748 2 LOCAL
5638 5749 2     ADDRESS,      : The address of the data
5639 5750 2     BIT_OFFSET,   : New bit offset from byte address
5640 5751 2     DTYPE;        : New dtype
5641 5752 2
5642 5753 2     ! Obtain the address.
5643 5754 2     !
5644 5755 2     ADDRESS = ..ARG_DESC[DSC$A_POINTER];
5645 5756 2
5646 5757 2     ! Obtain the bit offsets.
5647 5758 2     !
5648 5759 2     BIT_OFFSET = .OPERATOR [TOKEN$W_BIT_OFFSET];
5649 5760 2
5650 5761 2     ! Compute the new byte address.
5651 5762 2     !
5652 5763 2     ADDRESS = .ADDRESS + .BIT_OFFSET / 8;
5653 5764 2
5654 5765 2     ! Compute the bit offset. From it and the sign extension bit,
5655 5766 2     ! determine the new class and dtype.
5656 5767 2     !
5657 5768 2     BIT_OFFSET = .BIT_OFFSET MOD 8;
5658 5769 2     IF .BIT_OFFSET EQ 0
5659 5770 2     THEN
5660 5771 2         BEGIN
5661 5772 2             IF .OPERATOR [TOKEN$V_SGNEXT]
5662 5773 2             THEN
5663 5774 2                 DTYPE = DSC$K_DTYPE_SV
```

```
5664      ELSE
5665      DTYPE = DSC$K_DTYPE_V;
5666      SAVE_RESULT_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
5667      END
5668  ELSE
5669  BEGIN
5670  IF .OPERATOR [TOKEN$V_SGNEXT]
5671  THEN
5672  DTYPE = DSC$K_DTYPE_SVU
5673  ELSE
5674  DTYPE = DSC$K_DTYPE_VU;
5675  SAVE_RESULT_DESC[DSC$B_CLASS] = DSC$K_CLASS_UBS;
5676  END;
5677
5678  ! Save away the new dtype, bit offset, and length.
5679
5680  SAVE_RESULT_DESC [DSC$B_DTYPE] = .DTYPE;
5681  SAVE_RESULT_DESC [DSC$W_LENGTH] = .OPERATOR [TOKEN$W_BIT_LENGTH];
5682  SAVE_RESULT_DESC [DSC$A_POINTER] = .ADDRESS;
5683  SAVE_RESULT_DESC [DSC$L_POS] = .BIT_OFFSET;
5684
5685  ! Fill in the new address.
5686
5687  .RESULT_DESC[DSC$A_POINTER] = .ADDRESS;
5688
5689  ! Set a flag saying we've done a BLISS bit selection.
5690
5691  BLISS_BITSELECTION_FLAG1 = TRUE;
5692  END;
```

|    |    |    |            |        |                                     |      |
|----|----|----|------------|--------|-------------------------------------|------|
|    |    |    | 001C 00000 | .ENTRY | DBG\$BLISS BITSELECT, Save R2,R3,R4 | 5718 |
|    |    | 54 | 00000000'  | MOVAB  | SAVE_RESULT_DESC+3, R4              |      |
|    |    | 50 | 08         | MOVL   | ARG_DESC, R0                        | 5755 |
|    |    | 53 | 04         | MOVL   | @4(R0), ADDRESS                     |      |
|    |    | 51 | 04         | MOVL   | OPERATOR, R1                        | 5759 |
|    |    | 50 | 08         | MOVZWL | 8(R1), BIT_OFFSET                   |      |
|    | 52 | 50 | 08         | DIVL3  | #8, BIT_OFFSET, R2                  | 5763 |
|    |    | 53 | 52         | ADDL2  | R2, ADDRESS                         |      |
| 7E | 00 | 50 | 01         | EMUL   | #1, BIT_OFFSET, #0, -(SP)           | 5768 |
| 50 | 50 | 8E | 08         | EDIV   | #8, (SPT+ BIT_OFFSET, BIT_OFFSET    |      |
|    |    |    | 50         | TSTL   | BIT_OFFSET                          | 5769 |
|    |    |    | 11         | BNEQ   | 3\$                                 |      |
|    | 05 | 61 | 0A         | BBC    | #10, (R1), 1\$                      | 5772 |
|    |    | 52 | 29         | MOVL   | #41, DTYPE                          | 5774 |
|    |    |    | 03         | BRB    | 2\$                                 |      |
|    |    | 52 | 01         | MOVL   | #1, DTYPE                           | 5776 |
|    |    | 64 | 01         | MOVB   | #1, SAVE_RESULT_DESC+3              | 5777 |
|    |    |    | 0F         | BRB    | 6\$                                 | 5769 |
|    | 05 | 61 | 0A         | BBC    | #10, (R1), 4\$                      | 5781 |
|    |    | 52 | 2A         | MOVL   | #42, DTYPE                          | 5783 |
|    |    |    | 03         | BRB    | 5\$                                 |      |
|    |    | 52 | 22         | MOVL   | #34, DTYPE                          | 5785 |
|    |    | 64 | 0D         | MOVB   | #13, SAVE_RESULT_DESC+3             | 5786 |

DBGVALOP  
V04-000

7  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGVALOP.B32;1

Page 154  
(23)

|    |    |    |    |    |       |     |      |                                |   |      |
|----|----|----|----|----|-------|-----|------|--------------------------------|---|------|
| FF | A4 |    | 52 | 90 | 0004E | 68: | MOVB | DTYPE, SAVE_RESULT_DESC+2      | : | 5791 |
| FD | A4 | 0A | A1 | B0 | 00052 |     | MOVW | 10(R1), SAVE_RESULT_DESC       | : | 5792 |
| 01 | A4 |    | 53 | D0 | 00057 |     | MOVL | ADDRESS, SAVE_RESULT_DESC+4    | : | 5793 |
| 05 | A4 |    | 50 | D0 | 0005B |     | MOVL | BIT_OFFSET, SAVE_RESULT_DESC+8 | : | 5794 |
|    | 51 | 0C | AC | D0 | 0005F |     | MOVL | RESULT_DESC, R1                | : | 5798 |
| 04 | B1 |    | 53 | D0 | 00063 |     | MOVL | ADDRESS, @4(R1)                | : |      |
| D1 | A4 |    | 01 | D0 | 00067 |     | MOVL | #1, BLISS_BITSELECTION_FLAG1   | : | 5802 |
|    |    |    |    | 04 | 0006B |     | RET  |                                | : | 5803 |

; Routine Size: 108 bytes, Routine Base: DBG\$CODE + 0005

```
5694 5804 1 GLOBAL ROUTINE DBG$BLISS_INDIRECTION (ARG_DESC) =
5695 5805 1
5696 5806 1 FUNCTION
5697 5807 1     This routine is called from DBG$PERFORM_OPERATOR to perform a
5698 5808 1     BLISS indirection. The routine just fetches the object
5699 5809 1     given by the VMS descriptor, and returns the object as a longword
5700 5810 1     value.
5701 5811 1
5702 5812 1 INPUTS
5703 5813 1     ARG_DESC - A VMS descriptor representing the argument
5704 5814 1
5705 5815 1 OUTPUTS
5706 5816 1     The object that the VMS descriptor points to is fetched.
5707 5817 1     This value is returned. If there was no read access to the
5708 5818 1     address in the descriptor, the NOACCESSR message is
5709 5819 1     signalled.
5710 5820 1
5711 5821 1
5712 5822 2 BEGIN
5713 5823 2 BUILTIN
5714 5824 2     PROBER;
5715 5825 2
5716 5826 2 LOCAL
5717 5827 2     VALDESC: REF DBG$VALDESC;           ! Pointer to a value descriptor
5718 5828 2
5719 5829 2     ! Turn the VMS desc into a value descriptor
5720 5830 2
5721 5831 2 IF .BLISS_BITSELECTION_FLAG2
5722 5832 2 THEN
5723 5833 2     VALDESC = DBG$MAKE_VAL_DESC (SAVE_RESULT_DESC, DBG$K_VALUE_DESC)
5724 5834 2 ELSE
5725 5835 2     VALDESC = DBG$MAKE_VAL_DESC (.ARG_DESC, DBG$K_VALUE_DESC);
5726 5836 2
5727 5837 2     ! If we have already done the fetch,
5728 5838 2     ! return the value in the descriptor.
5729 5839 2
5730 5840 2 IF .BLISS_INDIRECTION_FLAG OR .BLISS_BITSELECTION_FLAG2
5731 5841 2 THEN
5732 5842 2     RETURN .VALDESC[DBG$L_VALUE_VALUE0]
5733 5843 2
5734 5844 2 ELSE
5735 5845 2
5736 5846 2     ! Check for read access.
5737 5847 2     !
5738 5848 2     IF PROBER (%REF(0), %REF(1), .VALDESC[DBG$L_VALUE_VALUE0])
5739 5849 2     THEN
5740 5850 2         RETURN ..VALDESC[DBG$L_VALUE_VALUE0]
5741 5851 2
5742 5852 2     ELSE
5743 5853 2         SIGNAL (DBG$_NOACCESSR, 1, .VALDESC[DBG$L_VALUE_VALUE0])
5744 5854 2
5744 5854 1 END;
```

|           |           |    |    |       |        |                               |   |      |
|-----------|-----------|----|----|-------|--------|-------------------------------|---|------|
| 52        | 00000000' | EF | 9E | 00002 | MOVAB  | BLISS_BITSELECTION_FLAG2, R2  | : | 5831 |
| 09        |           | 62 | E9 | 00009 | BLBC   | BLISS_BITSELECTION_FLAG2, 1\$ | : | 5833 |
| 7E        | 7A        | 8F | 9A | 0000C | MOVZBL | #122, -(SP)                   | : |      |
|           | 28        | A2 | 9F | 00010 | PUSHAB | SAVE_RESULT_DESC              | : |      |
|           |           | 07 | 11 | 00013 | BRB    | 2\$                           | : |      |
| 7E        | 7A        | 8F | 9A | 00015 | MOVZBL | #122, -(SP)                   | : | 5835 |
|           | 04        | AC | DD | 00019 | PUSHL  | ARG_DESC                      | : |      |
| 00000000G | 00        | 02 | FB | 0001C | CALLS  | #2, DBG\$MAKE_VAL_DESC        | : |      |
|           | 03        | A2 | E8 | 00023 | BLBS   | BLISS_INDIRECTION_FLAG, 3\$   | : | 5840 |
|           | 05        | 62 | E9 | 00027 | BLBC   | BLISS_BITSELECTION_FLAG2, 4\$ | : |      |
|           | 50        | A0 | D0 | 0002A | MOVL   | 32(VALDESC), R0               | : | 5842 |
|           |           |    | 04 | 0002E | RET    |                               | : |      |
| 20        | B0        | 01 | 00 | 0C    | PROBER | #0, #1, @32(VALDESC)          | : | 5848 |
|           |           |    | C5 | 13    | BEQL   | 5\$                           | : |      |
|           |           | 50 | B0 | D0    | MOVL   | @32(VALDESC), R0              | : | 5850 |
|           |           |    | 04 | 0003A | RET    |                               | : |      |
|           |           |    | A0 | DD    | PUSHL  | 32(VALDESC)                   | : | 5853 |
|           |           |    | 01 | DD    | PUSHL  | #1                            | : |      |
|           | 00028228  |    | 8F | DD    | PUSHL  | #164392                       | : |      |
| 00000000G | 00        | 03 | FB | 00046 | CALLS  | #3, LIB\$SIGNAL               | : |      |
|           |           |    | 04 | 0004D | RET    |                               | : | 5854 |

; Routine Size: 78 bytes, Routine Base: DBG\$CODE + 0071

```
5746 5855 1 GLOBAL ROUTINE DBG$CONV_TEXT_VALUE(VALUE1, VALUE2, NEW_TYPE) =
5747 5856 1
5748 5857 1 FUNCTION
5749 5858 1     Perform type conversion from an unconverted string to a value.
5750 5859 1     Note: this routine accepts unsigned dtype, and treat the
5751 5860 1     unsigned value as signed value (in other words, the T --> value
5752 5861 1     is signed.).
5753 5862 1
5754 5863 1 INPUTS
5755 5864 1     VALUE1 - Pointer to a value descriptor to be type-converted.
5756 5865 1     VALUE2 - Pointer to the target value descriptor.
5757 5866 1     NEW_TYPE- The eventual type of the value
5758 5867 1
5759 5868 1 OUTPUTS
5760 5869 1     Pointer to VALUE2 is returned.
5761 5870 1
5762 5871 2 BEGIN
5763 5872 2
5764 5873 2 MAP
5765 5874 2     VALUE1: REF DBG$VALDESC,      ! Pointer to Value Descr to convert
5766 5875 2     VALUE2: REF DBG$VALDESC;      ! Pointer to Target Descr
5767 5876 2
5768 5877 2 LOCAL
5769 5878 2     LENGTH,                      ! Length of the input data.
5770 5879 2     STATUS,                      ! Return code from library routines
5771 5880 2     STG_DESC: DBG$STG_DESC,      ! String descriptor for the input
5772 5881 2     STR_PTR,                     ! ASCII pointer
5773 5882 2     TARGET_DTYPE,               ! Target dtype
5774 5883 2     TARGET_LENGTH;              ! Target length
5775 5884 2
5776 5885 2
5777 5886 2 ! Fill in a string descriptor to be used as the source for the
5778 5887 2 ! conversion.
5779 5888 2
5780 5889 2 STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
5781 5890 2 STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
5782 5891 2 LENGTH = .VALUE1[DBG$W_VALUE_LENGTH];
5783 5892 2 STG_DESC[DSC$W_LENGTH] = .LENGTH;
5784 5893 2 STG_DESC[DSC$A_POINTER] = DBG$GET_TEMPMEM((3+.LENGTH)/4);
5785 5894 2
5786 5895 2 ! If language is ADA then we may have to strip underscores from the
5787 5896 2 ! text.
5788 5897 2
5789 5898 2 IF .DBG$GB_LANGUAGE EQL DBG$K_ADA
5790 5899 2 THEN
5791 5900 2     BEGIN
5792 5901 2         LOCAL
5793 5902 2             IN_PTR: REF VECTOR[ BYTE],
5794 5903 2             OUT_PTR: REF VECTOR[ BYTE];
5795 5904 2             IN_PTR = .VALUE1[DBG$W_VALUE_POINTER];
5796 5905 2             OUT_PTR = .STG_DESC[DSC$A_POINTER];
5797 5906 2             INCR I FROM 1 TO .LENGTH DO
5798 5907 2                 BEGIN
5799 5908 2                     IF .IN_PTR[0] NEQ '_' THEN
5800 5909 2                         BEGIN
5801 5910 2                             OUT_PTR[0] = .IN_PTR[0];
5802 5911 2                             OUT_PTR = .OUT_PTR + 1;
```

5803  
5804  
5805  
5806  
5807  
5808  
5809  
5810  
5811  
5812  
5813  
5814  
5815  
5816  
5817  
5818  
5819  
5820  
5821  
5822  
5823  
5824  
5825  
5826  
5827  
5828  
5829  
5830  
5831  
5832  
5833  
5834  
5835  
5836  
5837  
5838  
5839  
5840  
5841  
5842  
5843  
5844  
5845  
5846  
5847  
5848  
5849  
5850  
5851  
5852  
5853  
5854  
5855  
5856  
5857  
5858  
5859

```

END;
IN_PTR = .IN_PTR + 1;
END;
STG_DESC[DSC$W_LENGTH] = .OUT_PTR - .STG_DESC[DSC$A_POINTER];
END
ELSE
CH$MOVE(.LENGTH, .VALUE1[DBG$L_VALUE_POINTER], .STG_DESC[DSC$A_POINTER]);

! Case on the dtype to decide which conversion routine to call.
CASE .NEW_TYPE FROM DBG$K_MINIMUM_DTYPE TO DBG$K_MAXIMUM_DTYPE OF
SET

! Short integers. If the string input looked like an integer
! then convert it to longword.
[DSC$K_DTYPE_B, DSC$K_DTYPE_BU, DSC$K_DTYPE_W, DSC$K_DTYPE_WU,
DSC$K_DTYPE_L]:
BEGIN
IF .VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_INTEGER OR
.VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_HEX_INTEGER OR
.VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_BIN_INTEGER OR
.VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_OCT_INTEGER
THEN
TARGET_DTYPE = DSC$K_DTYPE_L
ELSE
TARGET_DTYPE = .VALUE1[DBG$B_VALUE_DTYPE];
END;

! If the target type is long integer and the input string looked
! like an integer then convert the string directly to the long
! integer. This ensures that we accept things like
! DEP/QUAD X = 1111111111
! If we first converted to longword and then to quad we would
! overflow.
[DSC$K_DTYPE_LU, DSC$K_DTYPE_Q, DSC$K_DTYPE_QU,
DSC$K_DTYPE_O, DSC$K_DTYPE_OU]:
BEGIN
IF .VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_INTEGER OR
.VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_HEX_INTEGER OR
.VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_BIN_INTEGER OR
.VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_OCT_INTEGER
THEN
TARGET_DTYPE = .NEW_TYPE
ELSE
TARGET_DTYPE = .VALUE1[DBG$B_VALUE_DTYPE];
END;

! If the target type is float, if the input string is in hex, bin,
! or oct radix, then hex, bin, oct always goes to Longword integer.
! For if we DEP f = Zhex 1, we want to convert text in hex to integer
! value, then converts 1 to float 1, then performs the DEP.

```

```
5860 5969 [DSCSK_DTYPE_F, DSCSK_DTYPE_D, DSCSK_DTYPE_G, DSCSK_DTYPE_H,  
5861 5970 DSCSK_DTYPE_P]:  
5862 5971 BEGIN  
5863 5972 IF .VALUE1[DBG$W_VALUE_TOKENCODE] NEQ TOKEN$K_HEX_INTEGER AND  
5864 5973 .VALUE1[DBG$W_VALUE_TOKENCODE] NEQ TOKEN$K_BIN_INTEGER AND  
5865 5974 .VALUE1[DBG$W_VALUE_TOKENCODE] NEQ TOKEN$K_OCT_INTEGER AND  
5866 5975 .VALUE1[DBG$W_VALUE_TOKENCODE] NEQ TOKEN$K_COMPLEMENT  
5867 5976 THEN  
5868 5977 TARGET_DTYPE = .NEW_TYPE  
5869 5978 ELSE  
5870 5979 TARGET_DTYPE = .VALUE1[DBG$B_VALUE_DTYPE];  
5871 5980 END;  
5872 5981  
5873 5982  
5874 5983 ! All the other types are either unsupported or fixed type conversion.  
5875 5984 ! For example, bit-string.  
5876 5985  
5877 5986 [INRANGE, OUTRANGE]:  
5878 5987 TARGET_DTYPE = .VALUE1[DBG$B_VALUE_DTYPE];  
5879 5988  
5880 5989 TES;  
5881 5990  
5882 5991 STR_PTR = UPLIT BYTE(%ASCIC '');  
5883 5992 CASE .VALUE1[DBG$W_VALUE_TOKENCODE] FROM TOKEN$K_MIN_OPERAND  
5884 5993 TO TOKEN$K_MAX_OPERAND OF  
5885 5994 SET  
5886 5995 [TOKEN$K_INTEGER]:  
5887 5996 STR_PTR = UPLIT BYTE(%ASCIC 'decimal');  
5888 5997  
5889 5998 [TOKEN$K_HEX_INTEGER]:  
5890 5999 STR_PTR = UPLIT BYTE(%ASCIC 'hexadecimal');  
5891 6000  
5892 6001 [TOKEN$K_FLOATING]:  
5893 6002 STR_PTR = UPLIT BYTE(%ASCIC 'f_float');  
5894 6003  
5895 6004 [TOKEN$K_EXP_E_FLOAT]:  
5896 6005 STR_PTR = UPLIT BYTE(%ASCIC 'f_float');  
5897 6006  
5898 6007 [TOKEN$K_EXP_D_FLOAT]:  
5899 6008 STR_PTR = UPLIT BYTE(%ASCIC 'd_float');  
5900 6009  
5901 6010 [TOKEN$K_EXP_Q_FLOAT]:  
5902 6011 STR_PTR = UPLIT BYTE(%ASCIC 'h_float');  
5903 6012  
5904 6013 [TOKEN$K_BIN_INTEGER]:  
5905 6014 STR_PTR = UPLIT BYTE(%ASCIC 'binary');  
5906 6015  
5907 6016 [TOKEN$K_OCT_INTEGER]:  
5908 6017 STR_PTR = UPLIT BYTE(%ASCIC 'octal');  
5909 6018  
5910 6019 [TOKEN$K_BIT_STRING]:  
5911 6020 STR_PTR = UPLIT BYTE(%ASCIC 'bit-string');  
5912 6021  
5913 6022 [TOKEN$K_PACK_DECIMAL]:  
5914 6023 STR_PTR = UPLIT BYTE(%ASCIC 'packed decimal');  
5915 6024  
5916 6025 [TOKEN$K_EXP_G_FLOAT]:
```

```
5917 6026 2
5918 6027 2
5919 6028 2
5920 6029 2
5921 6030 2
5922 6031 2
5923 6032 2
5924 6033 2
5925 6034 2
5926 6035 2
5927 6036 2
5928 6037 2
5929 6038 2
5930 6039 2
5931 6040 2
5932 6041 2
5933 6042 2
5934 6043 2
5935 6044 2
5936 6045 2
5937 6046 2
5938 6047 2
5939 6048 2
5940 6049 2
5941 6050 2
5942 6051 2
5943 6052 2
5944 6053 2
5945 6054 2
5946 6055 2
5947 6056 2
5948 6057 2
5949 6058 2
5950 6059 2
5951 6060 2
5952 6061 2
5953 6062 2
5954 6063 2
5955 6064 2
5956 6065 2
5957 6066 2
5958 6067 2
5959 6068 2
5960 6069 2
5961 6070 2
5962 6071 2
5963 6072 2
5964 6073 2
5965 6074 2
5966 6075 2
5967 6076 2
5968 6077 2
5969 6078 2
5970 6079 2
5971 6080 2
5972 6081 2
5973 6082 2
```

```
STR_PTR = UPLIT BYTE(ZASCIC 'g_float ');
[INRANGE, OTRANGE]:
0;
TES;

CASE .TARGET_DTYPE FROM DBGSK_MINIMUM_DTYPE
TO DBGSK_MAXIMUM_DTYPE OF
SET

Signed integers. Call a runtime routine to do the conversion
from text to integer.

Notes - known problems with this code:
1) In here, we treat the signed and unsigned
are the same, this may need to be fixed in the future to do the
the unsigned conversion. This only affects unsigned constants
between 2*31-1 and 2*32-1 (we treat these as signed longword,
so they overflow).
2) Another problem is that
the largest negative longword integer will overflow because
we are converting it to longword and then negating it at the end.

Since these problems only affect very large constants in
infrequent situations, they are being ignored at the present time.

[DSCSK_DTYPE_B, DSCSK_DTYPE_W, DSCSK_DTYPE_L,
DSCSK_DTYPE_BU, DSCSK_DTYPE_WU, DSCSK_DTYPE_LU]:
BEGIN
SELECTONE .TARGET_DTYPE OF
SET
[DSCSK_DTYPE_B, DSCSK_DTYPE_BU]:
TARGET_LENGTH = 1;

[DSCSK_DTYPE_W, DSCSK_DTYPE_WU]:
TARGET_LENGTH = 2;

[DSCSK_DTYPE_L, DSCSK_DTYPE_LU]:
TARGET_LENGTH = 4;

TES;

CASE .VALUE1[DBGSW_VALUE_TOKENCODE] FROM TOKENSK_MIN_OPERAND
TO TOKENSK_MAX_OPERAND OF
SET
[TOKENSK_INTEGER]:
BEGIN
STATUS = OTSSCVT_T1_L(STG_DESC,
.VALUE2[DBGSC_VALUE_POINTER], .TARGET_LENGTH);
END;

[TOKENSK_HEX_INTEGER]:
BEGIN
STATUS = OTSSCVT_T2_L(STG_DESC,
```

```
5974 6083 4
5975 6084 4
5976 6085 4
5977 6086 4
5978 6087 4
5979 6088 4
5980 6089 4
5981 6090 4
5982 6091 4
5983 6092 4
5984 6093 4
5985 6094 4
5986 6095 4
5987 6096 4
5988 6097 4
5989 6098 4
5990 6099 4
5991 6100 4
5992 6101 4
5993 6102 4
5994 6103 4
5995 6104 4
5996 6105 4
5997 6106 4
5998 6107 4
5999 6108 4
6000 6109 4
6001 6110 4
6002 6111 4
6003 6112 4
6004 6113 4
6005 6114 4
6006 6115 4
6007 6116 4
6008 6117 4
6009 6118 4
6010 6119 4
6011 6120 4
6012 6121 4
6013 6122 4
6014 6123 4
6015 6124 4
6016 6125 4
6017 6126 4
6018 6127 4
6019 6128 4
6020 6129 4
6021 6130 4
6022 6131 4
6023 6132 4
6024 6133 4
6025 6134 4
6026 6135 4
6027 6136 4
6028 6137 4
6029 6138 4
6030 6139 4
```

```
        .VALUE2[DBG$L_VALUE_POINTER], .TARGET_LENGTH);
    END;

[TOKEN$K_BIN_INTEGER]:
    BEGIN
        STATUS = OT$SCVT TO L(STG_DESC,
        .VALUE2[DBG$C_VALUE_POINTER], .TARGET_LENGTH);
    END;

[TOKEN$K_OCT_INTEGER]:
    BEGIN
        STATUS = OT$SCVT TO L(STG_DESC,
        .VALUE2[DBG$C_VALUE_POINTER], .TARGET_LENGTH);
    END;

[TOKEN$K_COMPLEMENT]:
    BEGIN
        STATUS = 1;
        .VALUE2[DBG$L_VALUE_POINTER] =
            MTH$JNOT(.STG_DESC[DSC$A_POINTER]);
    END;

[INRANGE, OUTRANGE]:
    $DBG_ERROR('DBGEVALOP\DBG$CONV_TEXT_VALUE');

TES;

IF NOT .STATUS
THEN
    BEGIN
        SELECT ONE .TARGET_DTYPE OF
        SET
        [DSC$K_DTYPE_B]:
            SIGNAL(DBG$UNACVT, 4, .STR_PTR, .LENGTH,
            .STG_DESC[DSC$A_POINTER],
            UPLIT BYTE (%ASCII 'byte integer'),
            .STATUS);

        [DSC$K_DTYPE_BU]:
            SIGNAL(DBG$UNACVT, 4, .STR_PTR, .LENGTH,
            .STG_DESC[DSC$A_POINTER],
            UPLIT BYTE (%ASCII 'byte logical'),
            .STATUS);

        [DSC$K_DTYPE_W]:
            SIGNAL(DBG$UNACVT, 4, .STR_PTR, .LENGTH,
            .STG_DESC[DSC$A_POINTER],
            UPLIT BYTE (%ASCII 'word integer'),
            .STATUS);

        [DSC$K_DTYPE_WU]:
            SIGNAL(DBG$UNACVT, 4, .STR_PTR, .LENGTH,
            .STG_DESC[DSC$A_POINTER],
            UPLIT BYTE (%ASCII 'word logical'),
            .STATUS);

        [DSC$K_DTYPE_L]:
```

```
6031 6140 4 SIGNAL (DBG$ UNACVT, 4, .STR_PTR, .LENGTH,  
6032 6141 4 .STG_DESC[DESC$A_POINTER],  
6033 6142 4 UPLIT BYTE (ASCII 'longword integer'),  
6034 6143 4 .STATUS);  
6035 6144 4  
6036 6145 4 [DSC$K_DTYPE_LU]:  
6037 6146 4 SIGNAL (DBG$ UNACVT, 4, .STR_PTR, .LENGTH,  
6038 6147 4 .STG_DESC[DESC$A_POINTER],  
6039 6148 4 UPLIT BYTE (ASCII 'longword logical'),  
6040 6149 4 .STATUS);  
6041 6150 4  
6042 6151 4 TES;  
6043 6152 4 END;  
6044 6153 4 VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;  
6045 6154 4 VALUE2[DBG$W_VALUE_LENGTH] = .TARGET_LENGTH;  
6046 6155 4 END;  
6047 6156 4  
6048 6157 4 [DSC$K_DTYPE_Q, DSC$K_DTYPE_QU]:  
6049 6158 4 BEGIN  
6050 6159 4 STATUS =  
6051 6160 4 DBG$CONV_TQUADWORD_VALUE(.VALUE1, .VALUE2);  
6052 6161 4 IF NOT .STATUS  
6053 6162 4 THEN  
6054 6163 4 BEGIN  
6055 6164 4 SELECTONE .TARGET_DTYPE OF  
6056 6165 4 SET  
6057 6166 4 [DSC$K_DTYPE_Q]:  
6058 6167 4 SIGNAL (DBG$ UNACVT, 4, .STR_PTR, .LENGTH,  
6059 6168 4 .STG_DESC[DESC$A_POINTER],  
6060 6169 4 UPLIT BYTE (ASCII 'quadword integer'),  
6061 6170 4 .STATUS);  
6062 6171 4  
6063 6172 4 [DSC$K_DTYPE_QU]:  
6064 6173 4 SIGNAL (DBG$ UNACVT, 4, .STR_PTR, .LENGTH,  
6065 6174 4 .STG_DESC[DESC$A_POINTER],  
6066 6175 4 UPLIT BYTE (ASCII 'quadword logical'),  
6067 6176 4 .STATUS);  
6068 6177 4  
6069 6178 4 TES;  
6070 6179 4 END;  
6071 6180 4 VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;  
6072 6181 4 VALUE2[DBG$W_VALUE_LENGTH] = 8;  
6073 6182 4 END;  
6074 6183 4  
6075 6184 4 [DSC$K_DTYPE_O, DSC$K_DTYPE_OU]:  
6076 6185 4 BEGIN  
6077 6186 4 STATUS =  
6078 6187 4 DBG$CONV_TOCTAWORD_VALUE(.VALUE1, .VALUE2);  
6079 6188 4 IF NOT .STATUS  
6080 6189 4 THEN  
6081 6190 4 BEGIN  
6082 6191 4 SELECTONE .TARGET_DTYPE OF  
6083 6192 4 SET  
6084 6193 4 [DSC$K_DTYPE_O]:  
6085 6194 4 SIGNAL (DBG$ UNACVT, 4, .STR_PTR, .LENGTH,  
6086 6195 4 .STG_DESC[DESC$A_POINTER],  
6087 6196 4
```

```

6088
6089
6090
6091
6092
6093
6094
6095
6096
6097
6098
6099
6100
6101
6102
6103
6104
6105
6106
6107
6108
6109
6110
6111
6112
6113
6114
6115
6116
6117
6118
6119
6120
6121
6122
6123
6124
6125
6126
6127
6128
6129
6130
6131
6132
6133
6134
6135
6136
6137
6138
6139
6140
6141
6142
6143
6144

```

```

        UPLIT BYTE (XASCIC 'octaword integer'),
        .STATUS);

[DSCSK DTYPE_OU]:
        SIGNAL(DBG$UNACVT, 4, .STR_PTR, .LENGTH,
        .STG_DESC[DSCSA_POINTER]);
        UPLIT BYTE (XASCIC 'octaword logical'),
        .STATUS);

        TES;
        END;

        VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;
        VALUE2[DBG$W_VALUE_LENGTH] = 16;
        END;

! Floating point. Call a runtime routine to do the conversion
! from text to float value.
[DSCSK DTYPE_F]:
        BEGIN
        STATUS = OTSSCVT_T_F(STG_DESC, .VALUE2[DBG$L_VALUE_POINTER]);
        IF NOT .STATUS
        THEN
                SIGNAL(DBG$UNACVT, 4, .STR_PTR, .LENGTH,
                .STG_DESC[DSCSA_POINTER]);
                UPLIT BYTE (XASCIC 'floating'),
                .STATUS);

                VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;
                VALUE2[DBG$W_VALUE_LENGTH] = 4;
                END;

! Double floating point. Call a runtime routine to do the conversion
! from text to double float value.
[DSCSK DTYPE_D]:
        BEGIN
        STATUS = OTSSCVT_T_D(STG_DESC, .VALUE2[DBG$L_VALUE_POINTER]);
        IF NOT .STATUS
        THEN
                SIGNAL(DBG$UNACVT, 4, .STR_PTR, .LENGTH,
                .STG_DESC[DSCSA_POINTER]);
                UPLIT BYTE (XASCIC 'double floating'),
                .STATUS);

                VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;
                VALUE2[DBG$W_VALUE_LENGTH] = 8;
                END;

! G Floating point. Call a runtime routine to do the conversion
! from text to G float value.
[DSCSK DTYPE_G]:
        BEGIN
        STATUS = OTSSCVT_T_G(STG_DESC, .VALUE2[DBG$L_VALUE_POINTER]);
        IF NOT .STATUS
        THEN

```

```
6145 6254 SIGNAL (DBG$UNACVT, 4, .STG_PTR, .LENGTH,  
6146 6255 .STG_DESC[DSC$A_POINTER],  
6147 6256 UPLIT BYTE (XASCII 'g_floating'),  
6148 6257 .STATUS);  
6149 6258  
6150 6259 VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;  
6151 6260 VALUE2[DBG$W_VALUE_LENGTH] = 8;  
6152 6261 END;  
6153 6262  
6154 6263 ! H Floating point. Call a runtime routine to do the conversion  
6155 6264 from text to H float value.  
6156 6265  
6157 6266 [DSC$K_DTYPE_H] :  
6158 6267 BEGIN  
6159 6268 STATUS = OT$SCVT_T_H(STG_DESC, .VALUE2[DBG$B_VALUE_POINTER]);  
6160 6269 IF NOT .STATUS  
6161 6270 THEN  
6162 6271 SIGNAL (DBG$UNACVT, 4, .STG_PTR, .LENGTH,  
6163 6272 .STG_DESC[DSC$A_POINTER],  
6164 6273 UPLIT BYTE (XASCII 'h_floating'),  
6165 6274 .STATUS);  
6166 6275  
6167 6276 VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;  
6168 6277 VALUE2[DBG$W_VALUE_LENGTH] = 16;  
6169 6278 END;  
6170 6279  
6171 6280 ! Pack decimal. Call a user routine to do the conversion  
6172 6281 from text to pack decimal.  
6173 6282  
6174 6283 [DSC$K_DTYPE_P] :  
6175 6284 VALUE2 = CONV_TEXT_PACK_VALUE(.VALUE1);  
6176 6285  
6177 6286 ! Text string. There is no conversion to be done here since the value  
6178 6287 is already in the right format.  
6179 6288  
6180 6289 [DSC$K_DTYPE_T] :  
6181 6290 0;  
6182 6291  
6183 6292 ! Bit-string. Call a Debug routine to do the conversion from text to  
6184 6293 bit-string format. The Debug routine actually calls a PL/I  
6185 6294 run-time routine to do the conversion; it is written in macro to  
6186 6295 simplify the interface (PL/I run-time routines notoriously do not  
6187 6296 adhere to the VAX calling standard). The bit-string format is thus  
6188 6297 PL/I specific.  
6189 6298  
6190 6299 [DSC$K_DTYPE_V] :  
6191 6300 BEGIN  
6192 6301 LOCAL  
6193 6302 STG_PTR: REF VECTOR[, BYTE];  
6194 6303  
6195 6304  
6196 6305 ! At present, allow only binary constants. Catch errors here,  
6197 6306 because PL/I run-time routines do not return a status.  
6198 6307  
6199 6308 STG_PTR = .STG_DESC[DSC$A_POINTER];  
6200 6309 INCR I FROM 0 TO .LENGTH-1 DO  
6201 6310 BEGIN
```

```
6202      6311      4      IF .STG_PTR[.I] NEQ %C'0' AND .STG_PTR[.I] NEQ %C'1'
6203      6312      4      THEN
6204      6313      4          SIGNAL (DBG$UNACVT, 4, .STR_PTR, .LENGTH,
6205      6314      4              .STG_DESC[DSC$A_POINTER],
6206      6315      4              UPLIT BYTE (%ASCII 'bit-string'), .STATUS);
6207      6316      4      END;
6208      6317      4
6209      6318      4      PLISCHARABIT R6(.STG_DESC[DSC$A_POINTER], .LENGTH,
6210      6319      4          .VALUE2[DBG$L_VALUE_POINTER], .VALUE2[DBG$W_VALUE_LENGTH]);
6211      6320      4      END;
6212      6321      4
6213      6322      4      ! We do not expect any other dtype, so signal an error if we see one.
6214      6323      4      !
6215      6324      4      [INRANGE, OTRANGE] :
6216      6325      4          $DBG_ERROR ('DBGEVALOP\DBG$CONV_TEXT_VALUE');
6217      6326      4
6218      6327      4      TES;
6219      6328      4
6220      6329      4      ! The conversion has been completed. Turn off the 'unconverted' bit in
6221      6330      4      ! the value descriptor. Return a pointer to the Value Descriptor which
6222      6331      4      ! contains the converted value.
6223      6332      4      !
6224      6333      4      VALUE2[DBG$V_DHDR_UNCVT] = 0;
6225      6334      4      VALUE2[DBG$W_VALUE_TOKENCODE] = 0;
6226      6335      4
6227      6336      4      ! Take care of the sign.
6228      6337      4      !
6229      6338      4      IF .VALUE1[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST OR
6230      6339      4          .VALUE1[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_POSCONST
6231      6340      4      THEN
6232      6341      4          BEGIN
6233      6342      4              IF .VALUE1[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST
6234      6343      4              THEN
6235      6344      4                  BEGIN
6236      6345      4                      IF .VALUE1[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST
6237      6346      4                      THEN
6238      6347      4                          BEGIN
6239      6348      4                              VALUE2[DBG$W_VALUE_SIGN_CODE] = 0;
6240      6349      4                              VALUE2 = DBG$EVAL [ANG_OPERATOR(DBG$GL_NEG_CONST_TOKEN,
6241      6350      4                                  .VALUE2, 0);
6242      6351      4                              END
6243      6352      4                          END
6244      6353      4                      ELSE
6245      6354      4                          BEGIN
6246      6355      4                              VALUE2[DBG$W_VALUE_SIGN_CODE] = 0;
6247      6356      4                              VALUE2 = DBG$EVAL [ANG_OPERATOR(DBG$GL_POS_CONST_TOKEN,
6248      6357      4                                  .VALUE2, 0);
6249      6358      4                              END;
6250      6359      4                          END;
6251      6360      4                      END;
6252      6361      4                      RETURN .VALUE2;
6253      6362      4                      END;
6254      6363      4                      L1:6315
6255      6364      4                      ! Referenced LOCAL symbol STATUS is probably not initialized
```

.PSECT DBG\$PLIT,NOWRT, SHR, PIC,0

|    |    |    |    |    |    |    |    |    |    |    |    |    |       |        |        |             |              |   |       |        |        |     |  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|--------|--------|-------------|--------------|---|-------|--------|--------|-----|--|
|    |    |    |    |    |    | 20 | 6C | 61 | 6D | 20 | 6C | 61 | 6D    | 69     | 63     | 65          | 64           | 00  | 05A74 | P.AKI: | .ASCII | <0> |  |
|    |    | 20 | 6C | 61 | 6D | 69 | 63 | 65 | 64 | 69 | 63 | 65 | 64    | 08     | 05A75  | P.AKJ:      | .ASCII       | <8>\decimal \                             |       |        |        |     |  |
|    |    |    |    |    |    | 20 | 74 | 61 | 6F | 6C | 66 | 5F | 66    | 0C     | 05A7E  | P.AKK:      | .ASCII       | <12>\hexadecimal \                        |       |        |        |     |  |
|    |    |    |    |    |    | 20 | 74 | 61 | 6F | 6C | 66 | 5F | 66    | 08     | 05A8B  | P.AKL:      | .ASCII       | <8>\f_float \                             |       |        |        |     |  |
|    |    |    |    |    |    | 20 | 74 | 61 | 6F | 6C | 66 | 5F | 66    | 08     | 05A94  | P.AKM:      | .ASCII       | <8>\f_float \                             |       |        |        |     |  |
|    |    |    |    |    |    | 20 | 74 | 61 | 6F | 6C | 66 | 5F | 66    | 08     | 05A9D  | P.AKN:      | .ASCII       | <8>\d_float \                             |       |        |        |     |  |
|    |    |    |    |    |    | 20 | 74 | 61 | 6F | 6C | 66 | 5F | 66    | 08     | 05AA6  | P.AKO:      | .ASCII       | <8>\h_float \                             |       |        |        |     |  |
|    |    |    |    |    |    | 20 | 79 | 72 | 61 | 6E | 69 | 62 | 07    | 05AAF  | P.AKP: | .ASCII      | <7>\binary \ |   |       |        |        |     |  |
|    |    |    |    |    |    | 20 | 6C | 61 | 74 | 63 | 6F | 06 | 05AB7 | P.AKQ: | .ASCII | <6>\octal \ |              |   |       |        |        |     |  |
| 6C | 61 | 6D | 20 | 67 | 6E | 69 | 72 | 74 | 73 | 2D | 74 | 69 | 62    | 0B     | 05ABE  | P.AKR:      | .ASCII       | <11>\bit-string \                         |       |        |        |     |  |
|    |    |    |    |    |    | 64 | 20 | 64 | 65 | 6B | 63 | 61 | 70    | 0F     | 05ACA  | P.AKS:      | .ASCII       | <15>\packed decimal \                     |       |        |        |     |  |
|    |    |    |    |    |    | 20 | 74 | 61 | 6F | 6C | 66 | 5F | 67    | 08     | 05AD9  |             |              |   |       |        |        |     |  |
| 24 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44    | 1D     | 05ADA  | P.AKT:      | .ASCII       | <8>\g_float \                             |       |        |        |     |  |
|    | 55 | 4C | 41 | 56 | 5F | 54 | 58 | 45 | 54 | 5F | 56 | 4E | 4F    | 43     | 05AE3  | P.AKU:      | .ASCII       | <29>\DBG\$VALOP\<92>\DBG\$CONV_TEXT_VALU\ |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 45     | 05AF2  |             |              |   |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 45     | 05B00  |             | .ASCII       | \E\                                       |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 0C     | 05B01  | P.AKV:      | .ASCII       | <12>\byte integer\                        |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 0C     | 05B0E  | P.AKW:      | .ASCII       | <12>\byte logical\                        |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 0C     | 05B1B  | P.AKX:      | .ASCII       | <12>\word integer\                        |       |        |        |     |  |
| 67 | 65 | 74 | 6E | 69 | 20 | 64 | 72 | 6F | 77 | 67 | 6E | 6F | 6C    | 10     | 05B28  | P.AKY:      | .ASCII       | <12>\word logical\                        |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 10     | 05B35  | P.AKZ:      | .ASCII       | <16>\longword integer\                    |       |        |        |     |  |
| 63 | 69 | 67 | 6F | 6C | 20 | 64 | 72 | 6F | 77 | 67 | 6E | 6F | 6C    | 10     | 05B44  |             |              |   |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 61     | 05B46  | P.ALA:      | .ASCII       | <16>\longword logical\                    |       |        |        |     |  |
| 67 | 65 | 74 | 6E | 69 | 20 | 64 | 72 | 6F | 77 | 64 | 61 | 75 | 71    | 10     | 05B55  |             |              |   |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 65     | 05B57  | P.ALB:      | .ASCII       | <16>\quadword integer\                    |       |        |        |     |  |
| 63 | 69 | 67 | 6F | 6C | 20 | 64 | 72 | 6F | 77 | 64 | 61 | 75 | 71    | 10     | 05B66  |             |              |   |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 61     | 05B68  | P.ALC:      | .ASCII       | <16>\quadword logical\                    |       |        |        |     |  |
| 67 | 65 | 74 | 6E | 69 | 20 | 64 | 72 | 6F | 77 | 61 | 74 | 63 | 6F    | 10     | 05B77  |             |              |   |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 65     | 05B79  | P.ALD:      | .ASCII       | <16>\octaword integer\                    |       |        |        |     |  |
| 63 | 69 | 67 | 6F | 6C | 20 | 64 | 72 | 6F | 77 | 61 | 74 | 63 | 6F    | 10     | 05B88  |             |              |   |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 61     | 05B8A  | P.ALE:      | .ASCII       | <16>\octaword logical\                    |       |        |        |     |  |
| 6E | 69 | 74 | 61 | 6F | 6C | 66 | 6E | 69 | 74 | 61 | 6F | 6C | 66    | 08     | 05B99  |             |              |   |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 0F     | 05B9B  | P.ALF:      | .ASCII       | <8>\floating\                             |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 67     | 05BA4  | P.ALG:      | .ASCII       | <15>\double floating\                     |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 0A     | 05BB3  |             |              |   |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 0A     | 05BB4  | P.ALH:      | .ASCII       | <10>\g_floating\                          |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 0A     | 05BBF  | P.ALI:      | .ASCII       | <10>\h_floating\                          |       |        |        |     |  |
| 24 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44    | 1D     | 05BCA  | P.ALJ:      | .ASCII       | <10>\bit-string\                          |       |        |        |     |  |
|    | 55 | 4C | 41 | 56 | 5F | 54 | 58 | 45 | 54 | 5F | 56 | 4E | 4F    | 43     | 05BD5  | P.ALK:      | .ASCII       | <29>\DBG\$VALOP\<92>\DBG\$CONV_TEXT_VALU\ |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 45     | 05BE4  |             |              |   |       |        |        |     |  |
|    |    |    |    |    |    |    |    |    |    |    |    |    |       | 45     | 05BF2  |             | .ASCII       | \E\                                       |       |        |        |     |  |

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

.ENTRY DBG\$CONV\_TEXT\_VALUE, Save R2,R3,R4,R5,R6,-  
R7,R8,R9,R10  
LIB\$SIGNAL, R10  
MOVAB P.AKI, R9  
SUBL2 #12, \$P  
MOVW #270, STG\_DESC+2  
MOVL VALUE1, R6

07FC 00000

|    |    |           |    |    |       |
|----|----|-----------|----|----|-------|
|    | 5A | 00000000G | 00 | 9E | 00002 |
|    | 59 | 00000000' | EF | 9E | 00009 |
|    | 5E |           | 0C | C2 | 00010 |
| 02 | AE | 010E      | 8F | B0 | 00013 |
|    | 56 | 04        | AC | D0 | 00019 |

5855

5890

5891

[illegible]

00128  
00128  
00133  
00138  
00143

5987  
5933  
5934  
5935  
5936  
5938  
5954  
5955  
5956  
5957  
5961  
5972  
5973  
5974  
5975  
5977  
5979  
5991  
5992



| Address | Hex | Op | Op2 | Op3 | Op4 | Op5 | Op6 | Op7 | Op8 | Op9 | Op10 | Op11 | Op12 | Op13 | Op14 | Op15 | Op16 | Op17 | Op18 | Op19 | Op20 | Op21 | Op22 | Op23 | Op24 | Op25 | Op26 | Op27 | Op28 | Op29 | Op30 | Op31 | Op32 | Op33 | Op34 | Op35 | Op36 | Op37 | Op38 | Op39 | Op40 | Op41 | Op42 | Op43 | Op44 | Op45 | Op46 | Op47 | Op48 | Op49 | Op50 | Op51 | Op52 | Op53 | Op54 | Op55 | Op56 | Op57 | Op58 | Op59 | Op60 | Op61 | Op62 | Op63 | Op64 | Op65 | Op66 | Op67 | Op68 | Op69 | Op70 | Op71 | Op72 | Op73 | Op74 | Op75 | Op76 | Op77 | Op78 | Op79 | Op80 | Op81 | Op82 | Op83 | Op84 | Op85 | Op86 | Op87 | Op88 | Op89 | Op90 | Op91 | Op92 | Op93 | Op94 | Op95 | Op96 | Op97 | Op98 | Op99 | Op100 | Op101 | Op102 | Op103 | Op104 | Op105 | Op106 | Op107 | Op108 | Op109 | Op110 | Op111 | Op112 | Op113 | Op114 | Op115 | Op116 | Op117 | Op118 | Op119 | Op120 | Op121 | Op122 | Op123 | Op124 | Op125 | Op126 | Op127 | Op128 | Op129 | Op130 | Op131 | Op132 | Op133 | Op134 | Op135 | Op136 | Op137 | Op138 | Op139 | Op140 | Op141 | Op142 | Op143 | Op144 | Op145 | Op146 | Op147 | Op148 | Op149 | Op150 | Op151 | Op152 | Op153 | Op154 | Op155 | Op156 | Op157 | Op158 | Op159 | Op160 | Op161 | Op162 | Op163 | Op164 | Op165 | Op166 | Op167 | Op168 | Op169 | Op170 | Op171 | Op172 | Op173 | Op174 | Op175 | Op176 | Op177 | Op178 | Op179 | Op180 | Op181 | Op182 | Op183 | Op184 | Op185 | Op186 | Op187 | Op188 | Op189 | Op190 | Op191 | Op192 | Op193 | Op194 | Op195 | Op196 | Op197 | Op198 | Op199 | Op200 | Op201 | Op202 | Op203 | Op204 | Op205 | Op206 | Op207 | Op208 | Op209 | Op210 | Op211 | Op212 | Op213 | Op214 | Op215 | Op216 | Op217 | Op218 | Op219 | Op220 | Op221 | Op222 | Op223 | Op224 | Op225 | Op226 | Op227 | Op228 | Op229 | Op230 | Op231 | Op232 | Op233 | Op234 | Op235 | Op236 | Op237 | Op238 | Op239 | Op240 | Op241 | Op242 | Op243 | Op244 | Op245 | Op246 | Op247 | Op248 | Op249 | Op250 | Op251 | Op252 | Op253 | Op254 | Op255 | Op256 | Op257 | Op258 | Op259 | Op260 | Op261 | Op262 | Op263 | Op264 | Op265 | Op266 | Op267 | Op268 | Op269 | Op270 | Op271 | Op272 | Op273 | Op274 | Op275 | Op276 | Op277 | Op278 | Op279 | Op280 | Op281 | Op282 | Op283 | Op284 | Op285 | Op286 | Op287 | Op288 | Op289 | Op290 | Op291 | Op292 | Op293 | Op294 | Op295 | Op296 | Op297 | Op298 | Op299 | Op300 | Op301 | Op302 | Op303 | Op304 | Op305 | Op306 | Op307 | Op308 | Op309 | Op310 | Op311 | Op312 | Op313 | Op314 | Op315 | Op316 | Op317 | Op318 | Op319 | Op320 | Op321 | Op322 | Op323 | Op324 | Op325 | Op326 | Op327 | Op328 | Op329 | Op330 | Op331 | Op332 | Op333 | Op334 | Op335 | Op336 | Op337 | Op338 | Op339 | Op340 | Op341 | Op342 | Op343 | Op344 | Op345 | Op346 | Op347 | Op348 | Op349 | Op350 | Op351 | Op352 | Op353 | Op354 | Op355 | Op356 | Op357 | Op358 | Op359 | Op360 | Op361 | Op362 | Op363 | Op364 | Op365 | Op366 | Op367 | Op368 | Op369 | Op370 | Op371 | Op372 | Op373 | Op374 | Op375 | Op376 | Op377 | Op378 | Op379 | Op380 | Op381 | Op382 | Op383 | Op384 | Op385 | Op386 | Op387 | Op388 | Op389 | Op390 | Op391 | Op392 | Op393 | Op394 | Op395 | Op396 | Op397 | Op398 | Op399 | Op400 | Op401 | Op402 | Op403 | Op404 | Op405 | Op406 | Op407 | Op408 | Op409 | Op410 | Op411 | Op412 | Op413 | Op414 | Op415 | Op416 | Op417 | Op418 |
|---------|-----|----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|---------|-----|----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|

|           |    |      |    |       |             |                     |            |      |
|-----------|----|------|----|-------|-------------|---------------------|------------|------|
|           | 18 | A0   | DD | 00261 | PUSHL       | 24(R0)              |            |      |
|           | 08 | AF   | 9F | 00264 | PUSHAB      | STG_DESC            | 6076       |      |
| 00000000G | 00 | 03   | FB | 00267 | CALLS       | #3, -OTSS\$CVT_T1_L |            |      |
|           |    | 3D   | 11 | 0026E | BRB         | 42\$                |            |      |
|           | 50 | 58   | DD | 00270 | 39\$: PUSHL | TARGET_LENGTH       | 6083       |      |
|           |    | AC   | DD | 00272 | MOVL        | VALUE2, R0          |            |      |
|           | 18 | A0   | DD | 00276 | PUSHL       | 24(R0)              |            |      |
|           | 08 | AF   | 9F | 00279 | PUSHAB      | STG_DESC            | 6082       |      |
| 00000000G | 00 | 03   | FB | 0027C | CALLS       | #3, -OTSS\$CVT_T2_L |            |      |
|           |    | 28   | 11 | 00283 | BRB         | 42\$                |            |      |
|           | 50 | 58   | DD | 00285 | 40\$: PUSHL | TARGET_LENGTH       | 6089       |      |
|           |    | AC   | DD | 00287 | MOVL        | VALUE2, R0          |            |      |
|           | 18 | A0   | DD | 0028B | PUSHL       | 24(R0)              |            |      |
|           | 08 | AF   | 9F | 0028E | PUSHAB      | STG_DESC            | 6088       |      |
| 00000000G | 00 | 03   | FB | 00291 | CALLS       | #3, -OTSS\$CVT_TB_L |            |      |
|           |    | 13   | 11 | 00298 | BRB         | 42\$                |            |      |
|           | 50 | 58   | DD | 0029A | 41\$: PUSHL | TARGET_LENGTH       | 6095       |      |
|           |    | AC   | DD | 0029C | MOVL        | VALUE2, R0          |            |      |
|           | 18 | A0   | DD | 002A0 | PUSHL       | 24(R0)              |            |      |
|           | 08 | AF   | 9F | 002A3 | PUSHAB      | STG_DESC            | 6094       |      |
| 00000000G | 00 | 03   | FB | 002A6 | CALLS       | #3, -OTSS\$CVT_TO_L |            |      |
|           | 52 | 50   | DD | 002AD | 42\$: MOVL  | R0, STATUS          |            |      |
|           |    | 15   | 11 | 002B0 | BRB         | 44\$                | 6071       |      |
|           | 52 | 01   | DD | 002B2 | 43\$: MOVL  | #1, STATUS          | 6100       |      |
|           | 54 | AC   | DD | 002B5 | MOVL        | VALUE2, R4          | 6101       |      |
|           |    | 04   | AE | DD    | 002B9       | PUSHL               | STG_DESC+4 | 6102 |
| 00000000G | 00 | 01   | FB | 002BC | CALLS       | #1, MTH\$JNOT       |            |      |
|           | 18 | 50   | DD | 002C3 | MOVL        | R0, @24(R4)         |            |      |
|           |    | 52   | E8 | 002C7 | 44\$: BLBS  | STATUS, 51\$        | 6110       |      |
|           | 06 | 55   | D1 | 002CA | CMPL        | TARGET_DTYPE, #6    | 6115       |      |
|           |    | 08   | 12 | 002CD | BNEQ        | 45\$                |            |      |
|           |    | 52   | DD | 002CF | PUSHL       | STATUS              | 6119       |      |
|           |    | 008D | C9 | 9F    | 002D1       | PUSHAB              | P.AKV      | 6118 |
|           |    | 3F   | 11 | 002D5 | BRB         | 50\$                | 6117       |      |
|           | 02 | 55   | D1 | 002D7 | 45\$: CMPL  | TARGET_DTYPE, #2    | 6121       |      |
|           |    | 08   | 12 | 002DA | BNEQ        | 46\$                |            |      |
|           |    | 52   | DD | 002DC | PUSHL       | STATUS              | 6125       |      |
|           |    | 009A | C9 | 9F    | 002DE       | PUSHAB              | P.AKW      | 6124 |
|           |    | 32   | 11 | 002E2 | BRB         | 50\$                | 6123       |      |
|           | 07 | 55   | D1 | 002E4 | 46\$: CMPL  | TARGET_DTYPE, #7    | 6127       |      |
|           |    | 08   | 12 | 002E7 | BNEQ        | 47\$                |            |      |
|           |    | 52   | DD | 002E9 | PUSHL       | STATUS              | 6131       |      |
|           |    | 00A7 | C9 | 9F    | 002EB       | PUSHAB              | P.AKX      | 6130 |
|           |    | 25   | 11 | 002EF | BRB         | 50\$                | 6129       |      |
|           | 03 | 55   | D1 | 002F1 | 47\$: CMPL  | TARGET_DTYPE, #3    | 6133       |      |
|           |    | 08   | 12 | 002F4 | BNEQ        | 48\$                |            |      |
|           |    | 52   | DD | 002F6 | PUSHL       | STATUS              | 6137       |      |
|           |    | 00B4 | C9 | 9F    | 002F8       | PUSHAB              | P.AKY      | 6136 |
|           |    | 18   | 11 | 002FC | BRB         | 50\$                | 6135       |      |
|           | 08 | 55   | D1 | 002FE | 48\$: CMPL  | TARGET_DTYPE, #8    | 6139       |      |
|           |    | 08   | 12 | 00301 | BNEQ        | 49\$                |            |      |
|           |    | 52   | DD | 00303 | PUSHL       | STATUS              | 6143       |      |
|           |    | 00C1 | C9 | 9F    | 00305       | PUSHAB              | P.AKZ      | 6142 |
|           |    | 08   | 11 | 00309 | BRB         | 50\$                | 6141       |      |
|           | 04 | 55   | D1 | 0030B | 49\$: CMPL  | TARGET_DTYPE, #4    | 6145       |      |
|           |    | 18   | 12 | 0030E | BNEQ        | 51\$                |            |      |
|           |    | 52   | DD | 00310 | PUSHL       | STATUS              | 6149       |      |

|           |    |          |      |       |       |        |                               |                   |      |
|-----------|----|----------|------|-------|-------|--------|-------------------------------|-------------------|------|
|           |    | 00D2     | C9   | 9F    | 00312 | PUSHAB | P.ALA                         | 6148              |      |
|           |    | 0C       | AE   | DD    | 00316 | PUSHL  | STG_DESC+4                    | 6147              |      |
|           |    | 0088     | 8F   | BB    | 00319 | PUSHR  | #^MZR3,R7>                    | 6146              |      |
|           |    |          | 04   | DD    | 0031D | PUSHL  | #4                            |                   |      |
|           |    | 00028E78 | 8F   | DD    | 0031F | PUSHL  | #167544                       |                   |      |
|           | 6A |          | 07   | FB    | 00325 | CALLS  | #7, LIB\$SIGNAL               |                   |      |
| 16        | 50 | 08       | AC   | DD    | 00328 | MOVL   | VALUE2, R0                    | 6153              |      |
| 14        | A0 |          | 55   | 90    | 0032C | MOVB   | TARGET_DTYPE, 22(R0)          |                   |      |
|           | A0 |          | 58   | 80    | 00330 | MOVW   | TARGET_LENGTH, 20(R0)         | 6154              |      |
|           |    |          | 0184 | 31    | 00334 | BRW    | 77\$                          | 6034              |      |
|           | 54 | 08       | AC   | DD    | 00337 | MOVL   | VALUE2, R4                    | 6160              |      |
|           |    |          | 54   | DD    | 0033B | PUSHL  | R4                            |                   |      |
|           |    |          | 56   | DD    | 0033D | PUSHL  | R6                            |                   |      |
| 0000V     | CF |          | 02   | FB    | 0033F | CALLS  | #2, DBG\$CONV_TQUADWORD_VALUE |                   |      |
|           | 52 |          | 50   | DD    | 00344 | MOVL   | R0, STATUS                    |                   |      |
|           | 03 |          | 52   | E9    | 00347 | BLBC   | STATUS, 54\$                  | 6161              |      |
|           |    | 00D3     | 31   | 0034A | 53\$: | BRW    | 67\$                          |                   |      |
|           | 09 |          | 55   | D1    | 0034D | 54\$:  | CMPL                          | TARGET_DTYPE, #9  | 6166 |
|           |    |          | 08   | 12    | 00350 | BNEQ   | 55\$                          |                   |      |
|           |    |          | 52   | DD    | 00352 | PUSHL  | STATUS                        | 6170              |      |
|           |    | 00E3     | C9   | 9F    | 00354 | PUSHAB | P.ALB                         | 6169              |      |
|           |    |          | 0B   | 11    | 00358 | BRB    | 56\$                          | 6168              |      |
|           | 05 |          | 55   | D1    | 0035A | 55\$:  | CMPL                          | TARGET_DTYPE, #5  | 6172 |
|           |    |          | EB   | 12    | 0035D | BNEQ   | 53\$                          |                   |      |
|           |    |          | 52   | DD    | 0035F | PUSHL  | STATUS                        | 6176              |      |
|           |    | 00F4     | C9   | 9F    | 00361 | PUSHAB | P.ALC                         | 6175              |      |
|           |    |          | 00A6 | 31    | 00365 | 56\$:  | BRW                           | 66\$              | 6174 |
|           | 54 | 08       | AC   | DD    | 00368 | 57\$:  | MOVL                          | VALUE2, R4        | 6188 |
|           |    |          | 54   | DD    | 0036C | PUSHL  | R4                            |                   |      |
|           |    |          | 56   | DD    | 0036E | PUSHL  | R6                            |                   |      |
| 0000V     | CF |          | 02   | FB    | 00370 | CALLS  | #2, DBG\$CONV_TOCTAWORD_VALUE |                   |      |
|           | 52 |          | 50   | DD    | 00375 | MOVL   | R0, STATUS                    |                   |      |
|           | 03 |          | 52   | E9    | 00378 | BLBC   | STATUS, 59\$                  | 6189              |      |
|           |    | 00DB     | 31   | 0037B | 58\$: | BRW    | 71\$                          |                   |      |
|           | 1A |          | 55   | D1    | 0037E | 59\$:  | CMPL                          | TARGET_DTYPE, #26 | 6194 |
|           |    |          | 08   | 12    | 00381 | BNEQ   | 60\$                          |                   |      |
|           |    |          | 52   | DD    | 00383 | PUSHL  | STATUS                        | 6198              |      |
|           |    | 0105     | C9   | 9F    | 00385 | PUSHAB | P.ALD                         | 6197              |      |
|           |    |          | 0B   | 11    | 00389 | BRB    | 61\$                          | 6196              |      |
|           | 19 |          | 55   | D1    | 0038B | 60\$:  | CMPL                          | TARGET_DTYPE, #25 | 6200 |
|           |    |          | EB   | 12    | 0038E | BNEQ   | 58\$                          |                   |      |
|           |    |          | 52   | DD    | 00390 | PUSHL  | STATUS                        | 6204              |      |
|           |    | 0116     | C9   | 9F    | 00392 | PUSHAB | P.ALE                         | 6203              |      |
|           |    |          | 00AE | 31    | 00396 | 61\$:  | BRW                           | 70\$              | 6202 |
|           | 54 | 08       | AC   | DD    | 00399 | 62\$:  | MOVL                          | VALUE2, R4        | 6217 |
|           |    | 18       | A4   | DD    | 0039D | PUSHL  | 24(R4)                        |                   |      |
|           |    | 04       | AE   | 9F    | 003A0 | PUSHAB | STG_DESC                      |                   |      |
| 00000000G | 00 |          | 02   | FB    | 003A3 | CALLS  | #2, -OTS\$CVT_T_F             |                   |      |
|           | 52 |          | 50   | DD    | 003AA | MOVL   | R0, STATUS                    |                   |      |
|           | 18 |          | 52   | EB    | 003AD | BLBS   | STATUS, 63\$                  | 6218              |      |
|           |    |          | 52   | DD    | 003B0 | PUSHL  | STATUS                        | 6223              |      |
|           |    | 0127     | C9   | 9F    | 003B2 | PUSHAB | P.ALF                         | 6222              |      |
|           |    | 0C       | AE   | DD    | 003B6 | PUSHL  | STG_DESC+4                    | 6221              |      |
|           |    | 0088     | 8F   | BB    | 003B9 | PUSHR  | #^MZR3,R7>                    | 6220              |      |
|           |    |          | 04   | DD    | 003BD | PUSHL  | #4                            |                   |      |
|           |    | 00028E78 | 8F   | DD    | 003BF | PUSHL  | #167544                       |                   |      |
|           | 6A |          | 07   | FB    | 003C5 | CALLS  | #7, LIB\$SIGNAL               |                   |      |

|           |    |          |      |    |       |      |        |                          |      |
|-----------|----|----------|------|----|-------|------|--------|--------------------------|------|
| 16        | A4 |          | 55   | 90 | 003C8 | 635: | MOVB   | TARGET DTYPE, 22(R4)     | 6225 |
| 14        | A4 |          | 04   | 80 | 003CC |      | MOVW   | #4, 20(R4)               | 6226 |
|           | 54 | 08       | 56   | 11 | 003D0 |      | BRB    | 685                      | 6034 |
|           |    | 18       | AC   | DD | 003D2 | 645: | MOVL   | VALUE2, R4               | 6234 |
|           |    | 04       | A4   | DD | 003D6 |      | PUSHL  | 24(R4)                   |      |
| 00000000G | 00 |          | AE   | 9F | 003D9 |      | PUSHAB | STG_DESC                 |      |
|           | 52 |          | 02   | FB | 003DC |      | CALLS  | #2, OTSSCVT_T_D          |      |
|           | 57 |          | 50   | DD | 003E3 |      | MOVL   | R0, STATUS               |      |
|           |    |          | 52   | E8 | 003E6 |      | BLBS   | STATUS, 678              | 6235 |
|           |    | 0130     | 52   | DD | 003E9 |      | PUSHL  | STATUS                   | 6240 |
|           |    |          | C9   | 9F | 003EB |      | PUSHAB | P.ALG                    | 6239 |
|           | 54 | 08       | 1D   | 11 | 003EF |      | BRB    | 665                      | 6238 |
|           |    | 18       | AC   | DD | 003F1 | 655: | MOVL   | VALUE2, R4               | 6251 |
|           |    | 04       | A4   | DD | 003F5 |      | PUSHL  | 24(R4)                   |      |
| 00000000G | 00 |          | AE   | 9F | 003F8 |      | PUSHAB | STG_DESC                 |      |
|           | 52 |          | 02   | FB | 003FB |      | CALLS  | #2, OTSSCVT_T_G          |      |
|           | 18 |          | 50   | DD | 00402 |      | MOVL   | R0, STATUS               |      |
|           |    |          | 52   | E8 | 00405 |      | BLBS   | STATUS, 678              | 6252 |
|           |    | 0140     | 52   | DD | 00408 |      | PUSHL  | STATUS                   | 6257 |
|           |    | 0C       | C9   | 9F | 0040A |      | PUSHAB | P.ALH                    | 6256 |
|           |    | 0088     | AE   | DD | 0040E | 665: | PUSHL  | STG_DESC+4               | 6255 |
|           |    |          | 8F   | BB | 00411 |      | PUSHR  | #*M2R3,R7>               | 6254 |
|           |    | 00028E78 | 04   | DD | 00415 |      | PUSHL  | #4                       |      |
|           | 6A |          | 8F   | DD | 00417 |      | PUSHL  | #167544                  |      |
| 16        | A4 |          | 07   | FB | 0041D |      | CALLS  | #7, LIBSSIGNAL           |      |
| 14        | A4 |          | 55   | 90 | 00420 | 675: | MOVB   | TARGET DTYPE, 22(R4)     | 6259 |
|           |    |          | 08   | 80 | 00424 |      | MOVW   | #8, 20(R4)               | 6260 |
|           | 54 | 08       | 44   | 11 | 00428 | 685: | BRB    | 735                      | 6034 |
|           |    | 18       | AC   | DD | 0042A | 695: | MOVL   | VALUE2, R4               | 6268 |
|           |    | 04       | A4   | DD | 0042E |      | PUSHL  | 24(R4)                   |      |
| 00000000G | 00 |          | AE   | 9F | 00431 |      | PUSHAB | STG_DESC                 |      |
|           | 52 |          | 02   | FB | 00434 |      | CALLS  | #2, OTSSCVT_T_H          |      |
|           | 18 |          | 50   | DD | 00438 |      | MOVL   | R0, STATUS               |      |
|           |    |          | 52   | E8 | 0043E |      | BLBS   | STATUS, 718              | 6269 |
|           |    | 0148     | 52   | DD | 00441 |      | PUSHL  | STATUS                   | 6274 |
|           |    | 0C       | C9   | 9F | 00443 |      | PUSHAB | P.ALI                    | 6273 |
|           |    | 0088     | AE   | DD | 00447 | 705: | PUSHL  | STG_DESC+4               | 6272 |
|           |    |          | 8F   | BB | 0044A |      | PUSHR  | #*M2R3,R7>               | 6271 |
|           |    | 00028E78 | 04   | DD | 0044E |      | PUSHL  | #4                       |      |
|           | 6A |          | 8F   | DD | 00450 |      | PUSHL  | #167544                  |      |
| 16        | A4 |          | 07   | FB | 00456 |      | CALLS  | #7, LIBSSIGNAL           |      |
| 14        | A4 |          | 55   | 90 | 00459 | 715: | MOVB   | TARGET DTYPE, 22(R4)     | 6276 |
|           |    |          | 10   | 80 | 0045D |      | MOVW   | #16, 20(R4)              | 6277 |
|           |    |          | 58   | 11 | 00461 |      | BRB    | 775                      | 6034 |
|           |    |          | 56   | DD | 00463 | 725: | PUSHL  | R6                       | 6284 |
| 0000V     | CF |          | 01   | FB | 00465 |      | CALLS  | #1, CONV TEXT_PACK_VALUE |      |
| 08        | AC |          | 50   | DD | 0046A |      | MOVL   | R0, VALUE2               |      |
|           |    |          | 48   | 11 | 0046E | 735: | BRB    | 775                      |      |
|           | 54 | 04       | AE   | DD | 00470 | 745: | MOVL   | STG_DESC+4, STG_PTR      | 6308 |
|           | 55 |          | 01   | CE | 00474 |      | MNEGL  | #1, -1                   | 6311 |
|           |    |          | 24   | 11 | 00477 |      | BRB    | 765                      |      |
|           | 30 |          | 6544 | 91 | 00479 | 755: | CMPB   | (1)[STG_PTR], #48        |      |
|           |    |          | 1E   | 13 | 0047D |      | BEQL   | 765                      |      |
|           | 31 |          | 6544 | 91 | 0047F |      | CMPB   | (1)[STG_PTR], #49        |      |
|           |    |          | 18   | 13 | 00483 |      | BEQL   | 765                      |      |
|           |    |          | 52   | DD | 00485 |      | PUSHL  | STATUS                   | 6315 |
|           |    | 0156     | C9   | 9F | 00487 |      | PUSHAB | P.ALJ                    |      |

|           |    |           |    |       |       |        |                            |      |
|-----------|----|-----------|----|-------|-------|--------|----------------------------|------|
|           |    | 0C        | AE | DD    | 00488 | PUSHL  | STG_DESC+4                 | 6314 |
|           |    | 0088      | 8F | BB    | 0048E | PUSHR  | #MZR3,R7>                  | 6313 |
|           |    |           | 04 | DD    | 00492 | PUSHL  | #4                         |      |
|           |    | 00028E78  | 8F | DD    | 00494 | PUSHL  | #167544                    |      |
| D8        | 6A |           | 07 | FB    | 0049A | CALLS  | #7, LIB\$SIGNAL            |      |
|           | 55 |           | 57 | F2    | 0049D | AOBLSS | LENGTH, 1, 75\$            | 6309 |
|           | 54 | 08        | AC | DD    | 004A1 | MOVL   | VALUE2, R4                 | 6319 |
|           | 53 | 14        | A4 | 3C    | 004A5 | MOVZWL | 20(R4), R3                 | 6318 |
|           | 52 | 18        | A4 | DD    | 004A9 | MOVL   | 24(R4), R2                 |      |
|           | 51 |           | 57 | DD    | 004AD | MOVL   | LENGTH, R1                 |      |
|           | 50 | 04        | AE | DD    | 004B0 | MOVL   | STG_DESC+4, R0             |      |
| 00000000G | 00 |           | 00 | FB    | 004B4 | CALLS  | #0, PLIS\$CHARABIT_R6      |      |
| 04        | A0 | 08        | AC | DD    | 004BB | MOVL   | VALUE2, R0                 | 6335 |
|           |    |           | 20 | 8A    | 004BF | BICB2  | #32, 4(R0)                 |      |
|           |    | 10        | A0 | B4    | 004C3 | CLRW   | 16(R0)                     | 6336 |
|           |    |           | 51 | D4    | 004C6 | CLRL   | R1                         | 6341 |
| 0042      | 8F | 12        | A6 | B1    | 004C8 | CMPL   | 18(R6), #66                |      |
|           |    |           | 04 | 12    | 004CE | BNEQ   | 78\$                       |      |
|           |    |           | 51 | D6    | 004D0 | INCL   | R1                         |      |
|           |    |           | 08 | 11    | 004D2 | BRB    | 79\$                       |      |
| 0043      | BF | 12        | A6 | B1    | 004D4 | CMPL   | 18(R6), #67                | 6342 |
|           |    |           | 28 | 12    | 004DA | BNEQ   | 82\$                       |      |
|           | 0F |           | 51 | E9    | 004DC | BLBC   | R1, 80\$                   | 6345 |
|           |    | 12        | A0 | B4    | 004DF | CLRW   | 18(R0)                     | 6348 |
|           |    |           | 7E | D4    | 004E2 | CLRL   | -(SP)                      | 6349 |
|           |    |           | 50 | DD    | 004E4 | PUSHL  | R0                         | 6350 |
|           |    | 00000000G | 00 | 9F    | 004E6 | PUSHAB | DBG\$GL_NEG_CONST_TOKEN    | 6349 |
|           |    |           | 0D | 11    | 004EC | BRB    | 81\$                       |      |
|           |    | 12        | A0 | B4    | 004EE | CLRW   | 18(R0)                     | 6355 |
|           |    |           | 7E | D4    | 004F1 | CLRL   | -(SP)                      | 6356 |
|           |    |           | 50 | DD    | 004F3 | PUSHL  | R0                         | 6357 |
|           |    | 00000000G | 00 | 9F    | 004F5 | PUSHAB | DBG\$GL_POS_CONST_TOKEN    | 6356 |
| 0000V     | CF |           | 03 | FB    | 004FB | CALLS  | #3, DBG\$VAL_LANG_OPERATOR |      |
| 08        | AC |           | 50 | DD    | 00500 | MOVL   | R0, VALUE2                 |      |
|           | 50 | 08        | AC | DD    | 00504 | MOVL   | VALUE2, R0                 | 6362 |
|           |    |           | 04 | 00508 | RET   |        |                            | 6364 |

; Routine Size: 1289 bytes, Routine Base: DBG\$CODE + 00BF

```
6257 6365 1 GLOBAL ROUTINE DBGS CONV_TQUADWORD_VALUE (VAL_DESC1, VAL_DESC2) =
6258 6366
6259 6367 FUNCTION
6260 6368     This routine takes in unconverted ascii string and converts it into
6261 6369     8 bytes Quadword value according to the given radix
6262 6370     (tokencode).
6263 6371
6264 6372 INPUTS
6265 6373     VAL_DESC1 - Pointer to a value descriptor to be converted.
6266 6374     VAL_DESC2 - Pointer to a value descriptor to be returned.
6267 6375
6268 6376 OUTPUTS
6269 6377     Overflow status is returned.
6270 6378
6271 6379 BEGIN
6272 6380
6273 6381 MAP
6274 6382     VAL_DESC1: REF DBGSVALDESC,      ! Pointer to value descriptor
6275 6383     VAL_DESC2: REF DBGSVALDESC;      ! Pointer to value descriptor
6276 6384
6277 6385 LOCAL
6278 6386     CHAR_VALUE: VECTOR[8, BYTE],      ! Value of the character
6279 6387     LENGTH,                          ! Length of the input data
6280 6388     PTR: REF VECTOR[BYTE],            ! Pointer to a vector of bytes
6281 6389     QUADWORD: VECTOR[8, BYTE],        ! 8 bytes of QUADWORD value
6282 6390     STATUS,                          ! Return status from conversion
6283 6391     STG_DESC: DBGSSTG_DESC;           ! String descriptor for signal
6284 6392
6285 6393     ! Prepare the string descriptor for signaling.
6286 6394     !
6287 6395     STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
6288 6396     STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
6289 6397     LENGTH = .VAL_DESC1[DBGS$W_VALUE_LENGTH];
6290 6398     STG_DESC[DSC$W_LENGTH] = .LENGTH;
6291 6399     STG_DESC[DSC$A_POINTER] = DBGSGET_TEMPHEM((3+.LENGTH)/4);
6292 6400     CHSMOVE(.STG_DESC[DSC$W_LENGTH], .VAL_DESC1[DBGS$L_VALUE_POINTER],
6293 6401     .STG_DESC[DSC$A_POINTER]);
6294 6402
6295 6403 INCR I FROM 0 TO 7 DO
6296 6404     BEGIN
6297 6405     QUADWORD[I] = 0;
6298 6406     CHAR_VALUE[I] = 0;
6299 6407     END;
6300 6408
6301 6409     ! Convert the value into 8 bytes one by one byte.
6302 6410     PTR = .VAL_DESC1[DBGS$L_VALUE_POINTER];
6303 6411     DECR I FROM .LENGTH TO -1 BY -1 DO
6304 6412     BEGIN
6305 6413     CASE .VAL_DESC1[DBGS$W_VALUE_TOKENCODE] FROM TOKEN$K_MIN_OPERAND
6306 6414     TO TOKEN$K_MAX_OPERAND OF
6307 6415     SET
6308 6416
6309 6417
6310 6418
6311 6419
6312 6420
6313 6421
```

```
6314
6315
6316
6317
6318
6319
6320
6321
6322
6323
6324
6325
6326
6327
6328
6329
6330
6331
6332
6333
6334
6335
6336
6337
6338
6339
6340
6341
6342
6343
6344
6345
6346
6347
6348
6349
6350
6351
6352
6353
6354
6355
6356
6357
6358
6359
6360
6361
6362
6363
6364
6365
6366
6367
6368
6369
6370
```

```
6422
6423
6424
6425
6426
6427
6428
6429
6430
6431
6432
6433
6434
6435
6436
6437
6438
6439
6440
6441
6442
6443
6444
6445
6446
6447
6448
6449
6450
6451
6452
6453
6454
6455
6456
6457
6458
6459
6460
6461
6462
6463
6464
6465
6466
6467
6468
6469
6470
6471
6472
6473
6474
6475
6476
6477
6478
```

```
! Validate legal 0-9 characters in decimal radix.
```

```
[TOKEN$K_INTEGER]:
```

```
  BEGIN
```

```
    IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 57
```

```
    THEN
```

```
      CHAR_VALUE[0] = .PTR[0] - 48
```

```
    ELSE
```

```
      SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
```

```
  END;
```

```
! Validate legal 0-9, A-F, a-f characters in hexadecimal radix.
```

```
[TOKEN$K_HEX_INTEGER]:
```

```
  BEGIN
```

```
    SELECT ONE .PTR[0] OF
```

```
      SET
```

```
        [48 TO 57]:
```

```
          CHAR_VALUE[0] = .PTR[0] - 48;
```

```
        [65 TO 70]:
```

```
          CHAR_VALUE[0] = .PTR[0] - 55;
```

```
        [97 TO 102]:
```

```
          CHAR_VALUE[0] = .PTR[0] - 87;
```

```
        [OTHERWISE]:
```

```
          SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
```

```
      TES;
```

```
  END;
```

```
! Validate legal 0-1 characters in binary radix.
```

```
[TOKEN$K_BIN_INTEGER]:
```

```
  BEGIN
```

```
    IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 49
```

```
    THEN
```

```
      CHAR_VALUE[0] = .PTR[0] - 48
```

```
    ELSE
```

```
      SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
```

```
  END;
```

```
! Validate legal 0-7 characters in octal radix.
```

```
[TOKEN$K_OCT_INTEGER]:
```

```
  BEGIN
```

```
    IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 55
```

```
    THEN
```

```
      CHAR_VALUE[0] = .PTR[0] - 48
```

```
    ELSE
```

```
      SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
```

```
  END;
```

```
[INRANGE, OUTRANGE]:
```

```
  $DBG_ERROR('DBGEVALOP\DBG$CONV_TQUADWORD_VALUE');
```

```

6371 6479
6372 6480
6373 6481
6374 6482
6375 6483
6376 6484
6377 6485
6378 6486
6379 6487
6380 6488
6381 6489
6382 6490
6383 6491
6384 6492
6385 6493
6386 6494
6387 6495
6388 6496
6389 6497
6390 6498
6391 6499
6392 6500
6393 6501
6394 6502
6395 6503
6396 6504
6397 6505
6398 6506
6399 6507
6400 6508
6401 6509
6402 6510

TES:
STATUS =
DBG$CVT_TQUADWORD_TO_VALUE(QUADWORD, CHAR_VALUE, .VAL_DESC1[DBG$W_VALUE_TOKENCODE]);
! Test overflow status.
IF NOT .STATUS
THEN
BEGIN
! If we could convert to an unsigned Quadword, do that with message.
STATUS =
DBG$CVT_TUQUADWORD_TO_VALUE(QUADWORD, CHAR_VALUE, .VAL_DESC1[DBG$W_VALUE_TOKENCODE]);
IF NOT .STATUS THEN EXITLOOP;
SIGNAL(DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME);
END;

PTR = .PTR + 1;
END;

! Install the value and fix up the value descriptor.
! CH$FILL(%'0', 16, .VAL_DESC1[DBG$L_VALUE_POINTER]);
! CH$MOVE(8, QUADWORD, .VAL_DESC2[DBG$L_VALUE_POINTER]);
! .VAL_DESC2 = .VAL_DESC1;
RETURN .STATUS;
END;
```

```

.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 22 05BF3 P.ALL: .ASCII '\DBGEVALOP\<92>\DBG$CONV_TQUADWORD_VALU\
5F 44 52 4F 57 44 41 55 51 54 5F 56 4E 4F 43 05C02
55 4C 41 45 05C11
45 05C15
.ASCII '\E\

.PSECT DBG$CODE,NOWRT, SHR, PIC,0
03FC 00000
59 00000000G 00 9E 00002
5E 1C C2 00009
02 AE 010E 8F B0 0000C
57 04 AC D0 00012
56 14 A7 3C 00016
6E 56 B0 0001A
50 03 A6 9E 0001D
7E 50 04 C7 00021
00000000G 00 01 FB 00025

MOVAB LIB$SIGNAL, R9
SUBL2 #28, SP
MOVW #270, STG_DESC+2
MOVL VAL_DESC1, R7
MOVZWL 20(R7), LENGTH
MOVW LENGTH, SIG_DESC
MOVAB 3(R6), R0
DIVL3 #4, R0, -(SP)
CALLS #1, DBG$GET_TEMPMEM
```

```

6365
6399
6400
6401
6402
```

| PC | Op | OpC | OpD | OpE | OpF | OpG | OpH | OpI | OpJ | OpK | OpL | OpM | OpN | OpO | OpP | OpQ | OpR | OpS | OpT | OpU | OpV | OpW | OpX | OpY | OpZ | OpAA | OpAB | OpAC | OpAD | OpAE | OpAF | OpAG | OpAH | OpAI | OpAJ | OpAK | OpAL | OpAM | OpAN | OpAO | OpAP | OpAQ | OpAR | OpAS | OpAT | OpAU | OpAV | OpAW | OpAX | OpAY | OpAZ | OpBA | OpBB | OpBC | OpBD | OpBE | OpBF | OpBG | OpBH | OpBI | OpBJ | OpBK | OpBL | OpBM | OpBN | OpBO | OpBP | OpBQ | OpBR | OpBS | OpBT | OpBU | OpBV | OpBW | OpBX | OpBY | OpBZ | OpCA | OpCB | OpCC | OpCD | OpCE | OpCF | OpCG | OpCH | OpCI | OpCJ | OpCK | OpCL | OpCM | OpCN | OpCO | OpCP | OpCQ | OpCR | OpCS | OpCT | OpCU | OpCV | OpCW | OpCX | OpCY | OpCZ | OpDA | OpDB | OpDC | OpDD | OpDE | OpDF | OpDG | OpDH | OpDI | OpDJ | OpDK | OpDL | OpDM | OpDN | OpDO | OpDP | OpDQ | OpDR | OpDS | OpDT | OpDU | OpDV | OpDW | OpDX | OpDY | OpDZ | OpEA | OpEB | OpEC | OpED | OpEE | OpEF | OpEG | OpEH | OpEI | OpEJ | OpEK | OpEL | OpEM | OpEN | OpEO | OpEP | OpEQ | OpER | OpES | OpET | OpEU | OpEV | OpEW | OpEX | OpEY | OpEZ | OpFA | OpFB | OpFC | OpFD | OpFE | OpFF | OpFG | OpFH | OpFI | OpFJ | OpFK | OpFL | OpFM | OpFN | OpFO | OpFP | OpFQ | OpFR | OpFS | OpFT | OpFU | OpFV | OpFW | OpFX | OpFY | OpFZ | OpGA | OpGB | OpGC | OpGD | OpGE | OpGF | OpGG | OpGH | OpGI | OpGJ | OpGK | OpGL | OpGM | OpGN | OpGO | OpGP | OpGQ | OpGR | OpGS | OpGT | OpGU | OpGV | OpGW | OpGX | OpGY | OpGZ | OpHA | OpHB | OpHC | OpHD | OpHE | OpHF | OpHG | OpHH | OpHI | OpHJ | OpHK | OpHL | OpHM | OpHN | OpHO | OpHP | OpHQ | OpHR | OpHS | OpHT | OpHU | OpHV | OpHW | OpHX | OpHY | OpHZ | OpIA | OpIB | OpIC | OpID | OpIE | OpIF | OpIG | OpIH | OpII | OpIJ | OpIK | OpIL | OpIM | OpIN | OpIO | OpIP | OpIQ | OpIR | OpIS | OpIT | OpIU | OpIV | OpIW | OpIX | OpIY | OpIZ | OpJA | OpJB | OpJC | OpJD | OpJE | OpJF | OpJG | OpJH | OpJI | OpJJ | OpJK | OpJL | OpJM | OpJN | OpJO | OpJP | OpJQ | OpJR | OpJS | OpJT | OpJU | OpJV | OpJW | OpJX | OpJY | OpJZ | OpKA | OpKB | OpKC | OpKD | OpKE | OpKF | OpKG | OpKH | OpKI | OpKJ | OpKK | OpKL | OpKM | OpKN | OpKO | OpKP | OpKQ | OpKR | OpKS | OpKT | OpKU | OpKV | OpKW | OpKX | OpKY | OpKZ | OpLA | OpLB | OpLC | OpLD | OpLE | OpLF | OpLG | OpLH | OpLI | OpLJ | OpLK | OpLL | OpLM | OpLN | OpLO | OpLP | OpLQ | OpLR | OpLS | OpLT | OpLU | OpLV | OpLW | OpLX | OpLY | OpLZ | OpMA | OpMB | OpMC | OpMD | OpME | OpMF | OpMG | OpMH | OpMI | OpMJ | OpMK | OpML | OpMM | OpMN | OpMO | OpMP | OpMQ | OpMR | OpMS | OpMT | OpMU | OpMV | OpMW | OpMX | OpMY | OpMZ | OpNA | OpNB | OpNC | OpND | OpNE | OpNF | OpNG | OpNH | OpNI | OpNJ | OpNK | OpNL | OpNM | OpNN | OpNO | OpNP | OpNQ | OpNR | OpNS | OpNT | OpNU | OpNV | OpNW | OpNX | OpNY | OpNZ | OpOA | OpOB | OpOC | OpOD | OpOE | OpOF | OpOG | OpOH | OpOI | OpOJ | OpOK | OpOL | OpOM | OpON | OpOO | OpOP | OpOQ | OpOR | OpOS | OpOT | OpOU | OpOV | OpOW | OpOX | OpOY | OpOZ | OpPA | OpPB | OpPC | OpPD | OpPE | OpPF | OpPG | OpPH | OpPI | OpPJ | OpPK | OpPL | OpPM | OpPN | OpPO | OpPP | OpPQ | OpPR | OpPS | OpPT | OpPU | OpPV | OpPW | OpPX | OpPY | OpPZ | OpQA | OpQB | OpQC | OpQD | OpQE | OpQF | OpQG | OpQH | OpQI | OpQJ | OpQK | OpQL | OpQM | OpQN | OpQO | OpQP | OpQQ | OpQR | OpQS | OpQT | OpQU | OpQV | OpQW | OpQX | OpQY | OpQZ | OpRA | OpRB | OpRC | OpRD | OpRE | OpRF | OpRG | OpRH | OpRI | OpRJ | OpRK | OpRL | OpRM | OpRN | OpRO | OpRP | OpRQ | OpRR | OpRS | OpRT | OpRU | OpRV | OpRW | OpRX | OpRY | OpRZ | OpSA | OpSB | OpSC | OpSD | OpSE | OpSF | OpSG | OpSH | OpSI | OpSJ |
|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|

|           |    |           |      |    |       |       |        |                                  |                       |      |
|-----------|----|-----------|------|----|-------|-------|--------|----------------------------------|-----------------------|------|
| 14        | AE | 62        | 07   | 1A | 000CF | 11\$: | BGTRU  | 13\$                             |                       |      |
|           |    |           | 30   | 83 | 000D1 | 12\$: | SUBB3  | #48, (PTR), CHAR_VALUE           | 6472                  |      |
|           |    |           | 0D   | 11 | 000D6 |       | BRB    | 15\$                             |                       |      |
|           |    |           | 5E   | DD | 000DB | 13\$: | PUSHL  | SP                               | 6474                  |      |
|           |    |           | 01   | DD | 000DA |       | PUSHL  | #1                               |                       |      |
|           |    | 000281B0  | BF   | DD | 000DC |       | PUSHL  | #164272                          |                       |      |
|           | 69 |           | 03   | FB | 000E2 | 14\$: | CALLS  | #3, LIB\$SIGNAL                  |                       |      |
|           | 7E | 10        | A7   | 3C | 000E3 | 15\$: | MOVZWL | 16(R7), -(SP)                    | 6483                  |      |
|           |    | 18        | AE   | 9F | 000E9 |       | PUSHAB | CHAR_VALUE                       |                       |      |
|           |    | 14        | AE   | 9F | 000EC |       | PUSHAB | QUADWORD                         |                       |      |
| 00000000G | 00 |           | 03   | FB | 000EF |       | CALLS  | #3, DBG\$CVT_TQUADWORD_TO_VALUE  |                       |      |
|           | 58 |           | 50   | D0 | 000F6 |       | MOVL   | R0, STATUS                       |                       |      |
|           | 28 |           | 58   | E8 | 000F9 |       | BLBS   | STATUS, 16\$                     | 6487                  |      |
|           | 7E | 10        | A7   | 3C | 000FC |       | MOVZWL | 16(R7), -(SP)                    | 6494                  |      |
|           |    | 18        | AE   | 9F | 00100 |       | PUSHAB | CHAR_VALUE                       |                       |      |
|           |    | 14        | AE   | 9F | 00103 |       | PUSHAB | QUADWORD                         |                       |      |
| 00000000G | 00 |           | 03   | FB | 00106 |       | CALLS  | #3, DBG\$CVT_TUQUADWORD_TO_VALUE |                       |      |
|           | 58 |           | 50   | D0 | 0010D |       | MOVL   | R0, STATUS                       |                       |      |
|           | 1B |           | 58   | E9 | 00110 |       | BLBC   | STATUS, 19\$                     | 6496                  |      |
|           |    | 00000000' | EF   | DD | 00113 |       | PUSHL  | DBG\$GL_OPCODE_NAME              | 6497                  |      |
|           |    |           | 01   | DD | 00119 |       | PUSHL  | #1                               |                       |      |
|           |    | 000286A3  | 8F   | DD | 0011B |       | PUSHL  | #165539                          |                       |      |
|           | 69 |           | 03   | FB | 00121 |       | CALLS  | #3, LIB\$SIGNAL                  |                       |      |
|           |    |           | 52   | D6 | 00124 | 16\$: | INCL   | PTR                              | 6500                  |      |
|           | 02 |           | 56   | F5 | 00126 | 17\$: | SOBGTR | 1, 18\$                          | 6417                  |      |
|           |    |           | 03   | 11 | 00129 |       | BRB    | 19\$                             |                       |      |
|           |    |           | FF1F | 31 | 0012B | 18\$: | BRW    | 2\$                              |                       |      |
| 18        | B0 | 0C        | 50   | 08 | AC    | 0012E | 19\$:  | MOVL                             | VAL_DESC2, R0         | 6507 |
|           |    |           | AE   | 08 | 28    | 00132 |        | MOVC3                            | #8, QUADWORD, @24(R0) |      |
|           |    |           | 50   | 58 | D0    | 00138 |        | MOVL                             | STATUS, R0            | 6509 |
|           |    |           |      | 04 | 0013B |       | RET    |                                  | 6510                  |      |

; Routine Size: 316 bytes, Routine Base: DBG\$CODE + 05C8

```
6404 6511 1 GLOBAL ROUTINE DBG$CONV_TOCTAWORD_VALUE(VAL_DESC1, VAL_DESC2) =
6405 6512 1
6406 6513 1 FUNCTION
6407 6514 1     This routine takes in unconverted ascii string and converts it into
6408 6515 1     16 bytes octaword value according to the given radix
6409 6516 1     (tokencode).
6410 6517 1
6411 6518 1 INPUTS
6412 6519 1     VAL_DESC1 - Pointer to a value descriptor to be converted.
6413 6520 1
6414 6521 1     VAL_DESC2 - Pointer to a value descriptor to be returned.
6415 6522 1
6416 6523 1 OUTPUTS
6417 6524 1     Overflow status is returned.
6418 6525 1
6419 6526 1
6420 6527 1 BEGIN
6421 6528 1
6422 6529 1 MAP
6423 6530 1     VAL_DESC1: REF DBG$VALDESC,      ! Pointer to value descriptor
6424 6531 1     VAL_DESC2: REF DBG$VALDESC;      ! Pointer to value descriptor
6425 6532 1
6426 6533 1 LOCAL
6427 6534 1     CHAR_VALUE: VECTOR[16, BYTE],    ! Value of the character
6428 6535 1     LENGTH,                          ! Length of the input data
6429 6536 1     PTR: REF VECTOR[ BYTE],          ! Pointer to a vector of bytes
6430 6537 1     OCTAWORD: VECTOR[16, BYTE],      ! 16 bytes of octaword value
6431 6538 1     STATUS,                          ! Return status from conversion
6432 6539 1     STG_DESC: DBG$STG_DESC;          ! String descriptor for signal
6433 6540 1
6434 6541 1
6435 6542 1 ! Prepare the string descriptor for signaling.
6436 6543 1
6437 6544 1 STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
6438 6545 1 STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
6439 6546 1 LENGTH = .VAL_DESC1[DBG$W_VALUE_LENGTH];
6440 6547 1 STG_DESC[DSC$W_LENGTH] = .LENGTH;
6441 6548 1 STG_DESC[DSC$A_POINTER] = DBG$GET_TEMPMEM((3+.LENGTH)/4);
6442 6549 1 CH$MOVE(.STG_DESC[DSC$W_LENGTH], .VAL_DESC1[DBG$W_VALUE_POINTER],
6443 6550 1     .STG_DESC[DSC$A_POINTER]);
6444 6551 1
6445 6552 1 INCR I FROM 0 TO 15 DO
6446 6553 1     BEGIN
6447 6554 1     OCTAWORD[I] = 0;
6448 6555 1     CHAR_VALUE[I] = 0;
6449 6556 1     END;
6450 6557 1
6451 6558 1
6452 6559 1 ! Convert the value into 16 bytes one by one byte.
6453 6560 1
6454 6561 1 PTR = .VAL_DESC1[DBG$W_VALUE_POINTER];
6455 6562 1 DECR I FROM .LENGTH TO -1 BY -1 DO
6456 6563 1     BEGIN
6457 6564 1     CASE .VAL_DESC1[DBG$W_VALUE_TOKENCODE] FROM TOKEN$K_MIN_OPERAND
6458 6565 1     TO     TOKEN$K_MAX_OPERAND OF
6459 6566 1
6460 6567 1         SET
```

```
6461 6568 3
6462 6569 3
6463 6570 3
6464 6571 3
6465 6572 4
6466 6573 4
6467 6574 4
6468 6575 4
6469 6576 4
6470 6577 4
6471 6578 3
6472 6579 3
6473 6580 3
6474 6581 3
6475 6582 3
6476 6583 3
6477 6584 4
6478 6585 4
6479 6586 4
6480 6587 4
6481 6588 4
6482 6589 4
6483 6590 4
6484 6591 4
6485 6592 4
6486 6593 4
6487 6594 4
6488 6595 4
6489 6596 4
6490 6597 4
6491 6598 3
6492 6599 3
6493 6600 3
6494 6601 3
6495 6602 4
6496 6603 4
6497 6604 4
6498 6605 4
6499 6606 4
6500 6607 4
6501 6608 3
6502 6609 3
6503 6610 3
6504 6611 3
6505 6612 3
6506 6613 3
6507 6614 4
6508 6615 4
6509 6616 4
6510 6617 4
6511 6618 4
6512 6619 4
6513 6620 3
6514 6621 3
6515 6622 3
6516 6623 3
6517 6624 3
```

```
! Validate legal 0-9 characters in decimal radix.
```

```
[TOKEN$K_INTEGER]:
```

```
  BEGIN
```

```
    IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 57
```

```
    THEN
```

```
      CHAR_VALUE[0] = .PTR[0] - 48
```

```
    ELSE
```

```
      SIGNAL(DBG$ INVNUMBER, 1, STG_DESC);
```

```
    END;
```

```
! Validate legal 0-9, A-F, a-f characters in hexadecimal radix.
```

```
[TOKEN$K_HEX_INTEGER]:
```

```
  BEGIN
```

```
    SELECT ONE .PTR[0] OF
```

```
      SET
```

```
        [48 TO 57]:
```

```
          CHAR_VALUE[0] = .PTR[0] - 48;
```

```
        [65 TO 70]:
```

```
          CHAR_VALUE[0] = .PTR[0] - 55;
```

```
        [97 TO 102]:
```

```
          CHAR_VALUE[0] = .PTR[0] - 87;
```

```
        [OTHERWISE]:
```

```
          SIGNAL(DBG$ INVNUMBER, 1, STG_DESC);
```

```
      TES;
```

```
    END;
```

```
! Validate legal 0-1 characters in binary radix.
```

```
[TOKEN$K_BIN_INTEGER]:
```

```
  BEGIN
```

```
    IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 49
```

```
    THEN
```

```
      CHAR_VALUE[0] = .PTR[0] - 48
```

```
    ELSE
```

```
      SIGNAL(DBG$ INVNUMBER, 1, STG_DESC);
```

```
    END;
```

```
! Validate legal 0-7 characters in octal radix.
```

```
[TOKEN$K_OCT_INTEGER]:
```

```
  BEGIN
```

```
    IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 55
```

```
    THEN
```

```
      CHAR_VALUE[0] = .PTR[0] - 48
```

```
    ELSE
```

```
      SIGNAL(DBG$ INVNUMBER, 1, STG_DESC);
```

```
    END;
```

```
[INRANGE, OUTRANGE]:
```

```
  $DBG_ERROR('DBGEVALOP\DBG$CONV_TOCTAWORD_VALUE');
```

```

6518      6625      TES;
6519      6626      STATUS =
6520      6627      DBG$CVT_TOCTAWORD_TO_VALUE(OCTAWORD, CHAR_VALUE, .VAL_DESC1[DBG$W_VALUE_TOKENCODE]);
6521      6628      ! Test overflow status.
6522      6629      ! IF NOT .STATUS THEN EXITLOOP;
6523      6630      PTR = .PTR + 1;
6524      6631      END;
6525      6632      ! Check for integer overflow on the sign bit.
6526      6633      ! IF .OCTAWORD[15] GEQ 128
6527      6634      THEN
6528      6635      SIGNAL(DBG$_INTOVF, 1, .DBG$GL_OPCODE_NAME);
6529      6636      ! Install the value and fix up the value descriptor.
6530      6637      CH$MOVE(16, OCTAWORD, .VAL_DESC2[DBG$L_VALUE_POINTER]);
6531      6638      RETURN .STATUS;
6532      6639      END;
6533      6640
6534      6641
6535      6642
6536      6643
6537      6644
6538      6645
6539      6646
6540      6647

```

```

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 22 05C16 P.ALM: .ASCII \DBGVALOP\<92>\DBG$CONV_TOCTAWORD_VALU
5F 44 52 4F 57 41 54 43 4F 54 5F 56 4E 4F 43 05C25
55 4C 41 56 05C34
45 05C38

```

```

.PSECT DBG$CODE, NOWRT, SHR, PIC, 0
.ENTRY DBG$CONV_TOCTAWORD_VALUE, Save R2, R3, R4, R5, -; 6511
MOVAB R6, R7, R8
LIB$SIGNAL, R8
SUBL2 #44, SP
MOVW #270, STG_DESC+2 6545
MOVL VAL_DESC1, R7 6546
MOVZWL 20(R7), LENGTH
MOVW LENGTH, STG_DESC 6547
MOVAB 3(R6), R2 6548
DIVL3 #4, R2, -(SP)
CALLS #1, DBG$GET_TEMPMEM
MOVL R0, STG_DESC+4
MOVC3 STG_DESC, @24(R7), @STG_DESC+4 6550
CLRL 1 6552
CLRB OCTAWORD[1] 6554
CLRB CHAR_VALUE[1] 6555
AOBLEQ #15, -1, 1$ 6552
MOVL 24(R7), PTR 6561
MOVAB 1(R6), 1 6564
BRW 16$

```

```

01FC 00000
58 00000000G 00 9E 00002
5E 2C C2 00009
02 AE 010E 8F B0 0000C
57 04 AC D0 00012
56 14 A7 3C 00016
6E 56 B0 0001A
52 03 A6 9E 0001D
52 04 C7 00021
7E 00000000G 00 01 FB 00025
04 BE 04 AE 50 D0 0002C
18 B7 6E 28 00030
50 D4 00036
0C AE40 94 00038 1$:
1C AE40 94 0003C
F4 50 0F F3 00040
53 18 A7 D0 00044
52 01 A6 9E 00048
00B1 31 0004C

```

| 0030 | 0F   | 01        | 10 | A7   | AF | 0004F | 2\$:  | CASEW  | 16(R7), #1, #15         |      |
|------|------|-----------|----|------|----|-------|-------|--------|-------------------------|------|
| 0020 | 0020 | 0020      |    | 0020 |    | 00054 | 3\$:  | .WORD  | 4\$-3\$,-               |      |
| 0020 | 0075 | 0068      |    | 0020 |    | 0005C |       |        | 4\$-3\$,-               |      |
| 0020 | 0020 | 0020      |    | 0020 |    | 00064 |       |        | 4\$-3\$,-               |      |
|      |      |           |    |      |    | 0006C |       |        | 5\$-3\$,-               |      |
|      |      |           |    |      |    |       |       |        | 6\$-3\$,-               |      |
|      |      |           |    |      |    |       |       |        | 4\$-3\$,-               |      |
|      |      |           |    |      |    |       |       |        | 4\$-3\$,-               |      |
|      |      |           |    |      |    |       |       |        | 4\$-3\$,-               |      |
|      |      |           |    |      |    |       |       |        | 4\$-3\$,-               |      |
|      |      |           |    |      |    |       |       |        | 4\$-3\$,-               |      |
|      |      |           |    |      |    |       |       |        | 9\$-3\$,-               |      |
|      |      |           |    |      |    |       |       |        | 10\$-3\$,-              |      |
|      |      |           |    |      |    |       |       |        | 4\$-3\$,-               |      |
|      |      |           |    |      |    |       |       |        | 4\$-3\$,-               |      |
|      |      |           |    |      |    |       |       |        | 4\$-3\$,-               |      |
|      |      |           |    |      |    |       |       |        | 4\$-3\$,-               |      |
|      |      |           |    |      |    |       |       |        | 4\$-3\$                 |      |
|      |      | 00000000' |    | EF   | 9F | 00074 | 4\$:  | PUSHAB | P.ALM                   | 6623 |
|      |      | 00028362  |    | 01   | DD | 0007A |       | PUSHL  | #1                      |      |
|      |      |           |    | 8F   | DD | 0007C |       | PUSHL  | #164706                 |      |
|      |      |           |    | 60   | 11 | 00082 |       | BRB    | 14\$                    |      |
|      | 30   |           |    | 63   | 91 | 00084 | 5\$:  | CMPB   | (PTR), #48              | 6573 |
|      |      |           |    | 51   | 1F | 00087 |       | BLSSU  | 13\$                    |      |
|      | 39   |           |    | 63   | 91 | 00089 |       | CMPB   | (PTR), #57              |      |
|      |      |           |    | 43   | 11 | 0008C |       | BRB    | 11\$                    |      |
|      | 30   |           |    | 63   | 91 | 0008E | 6\$:  | CMPB   | (PTR), #48              | 6587 |
|      |      |           |    | 05   | 1F | 00091 |       | BLSSU  | 7\$                     |      |
|      | 39   |           |    | 63   | 91 | 00093 |       | CMPB   | (PTR), #57              |      |
|      |      |           |    | 3B   | 1B | 00096 |       | BLEQU  | 12\$                    |      |
|      | 41   | 8F        |    | 63   | 91 | 00098 | 7\$:  | CMPB   | (PTR), #65              | 6589 |
|      |      |           |    | 0D   | 1F | 0009C |       | BLSSU  | 8\$                     |      |
|      | 46   | 8F        |    | 63   | 91 | 0009E |       | CMPB   | (PTR), #70              |      |
|      |      |           |    | 07   | 1A | 000A2 |       | BGTRU  | 8\$                     |      |
| 1C   | AE   |           |    | 63   | 83 | 000A4 |       | SUBB3  | #55, (PTR), CHAR_VALUE  | 6590 |
|      |      |           |    | 3C   | 11 | 000A9 |       | BRB    | 15\$                    |      |
|      | 61   | 8F        |    | 63   | 91 | 000AB | 8\$:  | CMPB   | (PTR), #97              | 6591 |
|      |      |           |    | 29   | 1F | 000AF |       | BLSSU  | 13\$                    |      |
|      | 66   | 8F        |    | 63   | 91 | 000B1 |       | CMPB   | (PTR), #102             |      |
|      |      |           |    | 23   | 1A | 000B5 |       | BGTRU  | 13\$                    |      |
| 1C   | AE   |           | A9 | 8F   | 81 | 000B7 |       | ADDB3  | #-87, (PTR), CHAR_VALUE | 6592 |
|      |      |           |    | 28   | 11 | 000BD |       | BRB    | 15\$                    |      |
|      | 30   |           |    | 63   | 91 | 000BF | 9\$:  | CMPB   | (PTR), #48              | 6603 |
|      |      |           |    | 16   | 1F | 000C2 |       | BLSSU  | 13\$                    |      |
|      | 31   |           |    | 63   | 91 | 000C4 |       | CMPB   | (PTR), #49              |      |
|      |      |           |    | 08   | 11 | 000C7 |       | BRB    | 11\$                    |      |
|      | 30   |           |    | 63   | 91 | 000C9 | 10\$: | CMPB   | (PTR), #48              | 6615 |
|      |      |           |    | 0C   | 1F | 000CC |       | BLSSU  | 13\$                    |      |
|      | 37   |           |    | 63   | 91 | 000CE |       | CMPB   | (PTR), #55              |      |
|      |      |           |    |      |    |       |       |        |                         |      |

|           |    |           |      |       |       |        |                                 |      |
|-----------|----|-----------|------|-------|-------|--------|---------------------------------|------|
| 00000000G | 00 | 14        | AE   | 9F    | 000EE | PUSHAB | OCTAWORD                        |      |
|           | 56 |           | 03   | FB    | 000F1 | CALLS  | #3, DBG\$CVT_TOCTAWORD_TO_VALUE |      |
|           | 0A |           | 50   | D0    | 000F8 | MOVL   | R0, STATUS                      | 6632 |
|           |    |           | 56   | E9    | 000FB | BLBC   | STATUS, 18\$                    | 6634 |
|           | 02 |           | 53   | D6    | 000FE | INCL   | PTR                             | 6562 |
|           |    |           | 52   | F5    | 00100 | SOBGTR | I, 17\$                         |      |
|           |    |           | 03   | 11    | 00103 | BRB    | 18\$                            |      |
|           |    |           | FF47 | 31    | 00105 | BRW    | 2\$                             |      |
| 80        | 8F | 1B        | AE   | 91    | 00108 | CMPB   | OCTAWORD+15, #128               | 6639 |
|           |    |           | 11   | 1F    | 0010D | BLSSU  | 19\$                            |      |
|           |    | 00000000' | EF   | DD    | 0010F | PUSHL  | DBG\$GL_OPCODE_NAME             | 6641 |
|           |    |           | 01   | DD    | 00115 | PUSHL  | #1                              |      |
|           |    | 000286A3  | 8F   | DD    | 00117 | PUSHL  | #165539                         |      |
|           | 68 |           | 03   | FB    | 0011D | CALLS  | #3, LIB\$SIGNAL                 |      |
|           | 50 | 08        | AC   | D0    | 00120 | MOVL   | VAL_DESC2, R0                   | 6645 |
| 18        | B0 | 0C        | AE   | 10    | 28    | MOVC3  | #16, OCTAWORD, @24(R0)          | 6646 |
|           | 50 |           | 56   | D0    | 0012A | MOVL   | STATUS, R0                      | 6647 |
|           |    |           | 04   | 0012D | RET   |        |                                 |      |

; Routine Size: 302 bytes, Routine Base: DBG\$CODE + 0704

```
6542 6648 1 GLOBAL ROUTINE DBG$CONV_TRFA_VALUE(VAL_DESC) =
6543 6649 1
6544 6650 1 FUNCTION
6545 6651 1     This routine takes in unconverted ascii string and converts it into
6546 6652 1     6 bytes Record File Address value according to the given radix
6547 6653 1     (tokencode).
6548 6654 1
6549 6655 1 INPUTS
6550 6656 1     VAL_DESC - Pointer to a value descriptor to be converted.
6551 6657 1
6552 6658 1 OUTPUTS
6553 6659 1     Converted value is returned in VAL_DESC.
6554 6660 1
6555 6661 1 BEGIN
6556 6662 1
6557 6663 1 MAP
6558 6664 1     VAL_DESC: REF DBG$VALDESC;      ! Pointer to value descriptor
6559 6665 1
6560 6666 1 LOCAL
6561 6667 1     CHAR_VALUE: VECTOR[6, BYTE],    ! Value of the character
6562 6668 1     LENGTH,                          ! Length of the input data
6563 6669 1     PTR: REF VECTOR[BYTE],          ! Pointer to a vector of bytes
6564 6670 1     RFA: VECTOR[6, BYTE],           ! 6 bytes of RFA value
6565 6671 1     STATUS,                          ! Return status from conversion
6566 6672 1     STG_DESC: DBG$STG_DESC;         ! String descriptor for signal
6567 6673 1
6568 6674 1 ! Won't accept the negative constant.
6569 6675 1 !
6570 6676 1 IF .VAL_DESC[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST
6571 6677 1 THEN
6572 6678 1     SIGNAL(DBG$CVTNEGUNS, 1, .DBG$GL_OPCODE_NAME);
6573 6679 1
6574 6680 1 ! Prepare the string descriptor for signaling.
6575 6681 1 !
6576 6682 1 STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
6577 6683 1 STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
6578 6684 1 LENGTH = .VAL_DESC[DBG$W_VALUE_LENGTH];
6579 6685 1 STG_DESC[DSC$B_LENGTH] = .LENGTH;
6580 6686 1 STG_DESC[DSC$B_POINTER] = DBG$GET_TEMPMEM((3+.LENGTH)/4);
6581 6687 1 CH$MOVE(.STG_DESC[DSC$W_LENGTH], .VAL_DESC[DBG$W_VALUE_POINTER],
6582 6688 1     .STG_DESC[DSC$B_POINTER]);
6583 6689 1
6584 6690 1 INCR I FROM 0 TO 5 DO
6585 6691 1     BEGIN
6586 6692 1         RFA[I] = 0;
6587 6693 1         CHAR_VALUE[I] = 0;
6588 6694 1     END;
6589 6695 1
6590 6696 1 ! Convert the value into 6 bytes one by one byte.
6591 6697 1 !
6592 6698 1 PTR = .VAL_DESC[DBG$W_VALUE_POINTER];
6593 6699 1
6594 6700 1 DECR I FROM .LENGTH TO 1 BY -1 DO
6595 6701 1
6596 6702 1
6597 6703 1
6598 6704 1
```

```
6599
6600
6601
6602
6603
6604
6605
6606
6607
6608
6609
6610
6611
6612
6613
6614
6615
6616
6617
6618
6619
6620
6621
6622
6623
6624
6625
6626
6627
6628
6629
6630
6631
6632
6633
6634
6635
6636
6637
6638
6639
6640
6641
6642
6643
6644
6645
6646
6647
6648
6649
6650
6651
6652
6653
6654
6655
```

```
6705
6706
6707
6708
6709
6710
6711
6712
6713
6714
6715
6716
6717
6718
6719
6720
6721
6722
6723
6724
6725
6726
6727
6728
6729
6730
6731
6732
6733
6734
6735
6736
6737
6738
6739
6740
6741
6742
6743
6744
6745
6746
6747
6748
6749
6750
6751
6752
6753
6754
6755
6756
6757
6758
6759
6760
6761
```

```
BEGIN
CASE .VAL_DESC[DBG$W_VALUE_TOKENCODE] FROM TOKEN$K_MIN_OPERAND
TO TOKEN$K_MAX_OPERAND OF
SET

! Validate legal 0-9 characters in decimal radix.
[TOKEN$K_INTEGER]:
BEGIN
IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 57
THEN
CHAR_VALUE[0] = .PTR[0] - 48
ELSE
SIGNAL(DBG$INVNUMBER, 1, STG_DESC);
END;

! Validate legal 0-9, A-F, a-f characters in hexadecimal radix.
[TOKEN$K_HEX_INTEGER]:
BEGIN
SELECTONE .PTR[0] OF
SET
[48 TO 57]:
CHAR_VALUE[0] = .PTR[0] - 48;
[65 TO 70]:
CHAR_VALUE[0] = .PTR[0] - 55;
[97 TO 102]:
CHAR_VALUE[0] = .PTR[0] - 87;
[OTHERWISE]:
SIGNAL(DBG$INVNUMBER, 1, STG_DESC);
TES;
END;

! Validate legal 0-1 characters in binary radix.
[TOKEN$K_BIN_INTEGER]:
BEGIN
IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 49
THEN
CHAR_VALUE[0] = .PTR[0] - 48
ELSE
SIGNAL(DBG$INVNUMBER, 1, STG_DESC);
END;

! Validate legal 0-7 characters in octal radix.
[TOKEN$K_OCT_INTEGER]:
BEGIN
IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 55
THEN
CHAR_VALUE[0] = .PTR[0] - 48
ELSE
SIGNAL(DBG$INVNUMBER, 1, STG_DESC);
```

```

6656
6657
6658
6659
6660
6661
6662
6663
6664
6665
6666
6667
6668
6669
6670
6671
6672
6673
6674
6675
6676
6677
6678
6679
6680
6681
6682
6683
6684
6685
6686
6687
6688
6689
6690
6691
6692

```

```

END;
[INRANGE, OUTRANGE]:
$DBG_ERROR('DBGEVALOP\DBG$CONV_TRFA_VALUE');
TES;
STATUS =
DBG$CVT_TRFA_TO_VALUE(RFA, CHAR_VALUE, .VAL_DESC[DBG$W_VALUE_TOKENCODE]);
! Test overflow status.
IF NOT .STATUS
THEN
BEGIN
SIGNAL(DBG$_IRFAOVF, 1, .DBG$GL_OPCODE_NAME);
EXITLOOP;
END;
PTR = .PTR + 1;
END;
! Install the value and fix up the value descriptor.
VAL_DESC[DBG$V_DHDR_UNCVT] = 0;
VAL_DESC[DBG$B_DHDR_FCODE] = RST$K_TYPE_RFA;
VAL_DESC[DBG$B_VALUE_CLASS] = 0;
VAL_DESC[DBG$B_VALUE_DTYPE] = 0;
VAL_DESC[DBG$W_VALUE_LENGTH] = 6;
CH$FILL('0', 16, .VAL_DESC[DBG$L_VALUE_POINTER]);
CH$MOVE(6, RFA, .VAL_DESC[DBG$L_VALUE_POINTER]);
VAL_DESC[DBG$W_VALUE_TOKENCODE] = 0;
VAL_DESC[DBG$W_VALUE_SIGN_CODE] = 0;
RETURN .VAL_DESC;
END;

```

```

.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 1D 05C39 P.ALN: .ASCII <29>\DBGEVALOP\<92>\DBG$CONV_TRFA_VALU
55 4C 41 56 5F 41 46 52 54 5F 56 4E 4F 43 05C48
45 05C56 .ASCII \E\

.PSECT DBG$CODE,NOWRT, SHR, PIC,0
03FC 00000
59 00000000' EF 9E 00002
58 00000000G 00 9E 00009
5E 1C C2 00010
57 04 AC D0 00013
0042 8F 12 A7 B1 00017
MOVAB DBG$GL_OPCODE_NAME, R9
MOVAB LIB$SIGNAL, R8
SUBL2 #28, SP
MOVL VAL_DESC, R7
CMPW 18(R7), #60

```

[illegible]

|    |    |           |    |    |    |       |        |                             |      |
|----|----|-----------|----|----|----|-------|--------|-----------------------------|------|
| 14 | AE | 66        | 8F | 62 | 91 | 000CB | CMPB   | (PTR), #102                 | 6734 |
|    |    |           |    | 23 | 1A | 000CF | BGTRU  | 14\$                        |      |
|    |    |           | 62 | 8F | 81 | 000D1 | ADDB3  | #-87, (PTR), CHAR_VALUE     | 6745 |
|    |    |           | 30 | 28 | 11 | 000D7 | BRB    | 16\$                        |      |
|    |    |           |    | 62 | 91 | 000D9 | CMPB   | (PTR), #48                  | 6757 |
|    |    |           | 31 | 16 | 1F | 000DC | BLSSU  | 14\$                        |      |
|    |    |           |    | 62 | 91 | 000DE | CMPB   | (PTR), #49                  |      |
|    |    |           | 30 | 08 | 11 | 000E1 | BRB    | 12\$                        |      |
|    |    |           | 37 | 62 | 91 | 000E3 | CMPB   | (PTR), #48                  | 6759 |
|    |    |           |    | 0C | 1F | 000E6 | BLSSU  | 14\$                        |      |
|    |    |           |    | 62 | 91 | 000E8 | CMPB   | (PTR), #55                  | 6761 |
| 14 | AE |           | 62 | 07 | 1A | 000EB | BGTRU  | 14\$                        |      |
|    |    |           |    | 30 | 83 | 000ED | SUBB3  | #48, (PTR), CHAR_VALUE      | 6770 |
|    |    |           |    | 0D | 11 | 000F2 | BRB    | 16\$                        |      |
|    |    |           |    | 5E | DD | 000F4 | PUSHL  | SP                          |      |
|    |    |           |    | 01 | DD | 000F6 | PUSHL  | #1                          |      |
|    |    |           |    | 8F | DD | 000F8 | PUSHL  | #164272                     |      |
|    |    | 000281B0  |    | 03 | FB | 000FE | CALLS  | #3, LIB\$SIGNAL             |      |
|    |    | 68        |    | A7 | 3C | 00101 | MOVZWL | 16(R7), -(SP)               | 6775 |
|    |    | 7E        | 10 | AE | 9F | 00105 | PUSHAB | CHAR_VALUE                  |      |
|    |    |           | 18 | AE | 9F | 00108 | PUSHAB | RFA                         |      |
|    |    |           | 14 | 03 | FB | 0010B | CALLS  | #3, DBG\$CVT_TRFA_TO_VALUE  |      |
|    |    | 00000000G | 00 | 50 | D0 | 00112 | MOVL   | R0, STATUS                  | 6778 |
|    |    |           | 53 | 53 | E8 | 00115 | BLBS   | STATUS, 17\$                |      |
|    |    |           | OF | 69 | DD | 00118 | PUSHL  | DBG\$GL_OPCODE_NAME         |      |
|    |    |           |    | 01 | DD | 0011A | PUSHL  | #1                          |      |
|    |    |           |    | 8F | DD | 0011C | PUSHL  | #167883                     |      |
|    |    | 00028FCB  |    | 03 | FB | 00122 | CALLS  | #3, LIB\$SIGNAL             |      |
|    |    | 68        |    | 0A | 11 | 00125 | BRB    | 20\$                        | 6777 |
|    |    |           |    | 52 | D6 | 00127 | INCL   | PTR                         | 6782 |
|    |    |           | 02 | 56 | F5 | 00129 | SOBGTR | 1, 19\$                     | 6704 |
|    |    |           |    | 03 | 11 | 0012C | BRB    | 20\$                        |      |
|    |    |           |    | FF | 38 | 0012E | BRW    | 3\$                         |      |
|    |    |           |    | 20 | 8A | 00131 | BICB2  | #32, 4(R7)                  | 6788 |
|    |    | 04        | A7 | 14 | 90 | 00135 | MOVB   | #20, 6(R7)                  | 6789 |
|    |    | 06        | A7 | 06 | D0 | 00139 | MOVL   | #6, 20(R7)                  | 6792 |
| 10 |    | 30        | 6E | 00 | 2C | 0013D | MOVC5  | #0, (SP), #48, #16, @24(R7) | 6793 |
|    |    |           |    | B7 |    | 00142 |        |                             |      |
|    |    |           |    | 06 | 28 | 00144 | MOVC3  | #6, RFA, @24(R7)            | 6794 |
|    |    |           |    | A7 | D4 | 0014A | CLRL   | 16(R7)                      | 6795 |
|    |    |           |    | 50 | D0 | 0014D | MOVL   | R7, R0                      | 6797 |
|    |    |           |    | 57 | 04 | 00150 | RET    |                             | 6798 |

; Routine Size: 337 bytes, Routine Base: DBG\$CODE + 0832

```
6694 6799 1
6695 6800 1 GLOBAL ROUTINE DBG$DO_MAPPING (LEFT_ARG) : NOVALUE =
6696 6801 1
6697 6802 1 FUNCTION
6698 6803 1     Perform a language-specific "type mapping" on the given
6699 6804 1     value descriptor. For example, in FORTRAN, we map unsigned integer
6700 6805 1     types to signed integer types, because the compiler is given us
6701 6806 1     the BU, WU, or LU types, while the language really does signed
6702 6807 1     arithmetic on these.
6703 6808 1
6704 6809 1 INPUTS
6705 6810 1     LEFT_ARG      - points to the value descriptor to be mapped.
6706 6811 1
6707 6812 1 OUTPUTS
6708 6813 1     The value descriptor pointed to by LEFT_ARG may be modified.
6709 6814 1
6710 6815 2 BEGIN
6711 6816 2 MAP
6712 6817 2     LEFT_ARG: REF DBG$VALDESC;
6713 6818 2
6714 6819 2 LOCAL
6715 6820 2     COMPLIST: REF VECTOR[.LONG],
6716 6821 2     DSTPTR,
6717 6822 2     LEFT_TYPE,                ! Dtype for original left operand type
6718 6823 2     MAKE_BOOLEAN_FLAG,
6719 6824 2     MAP_TBL_ENTRY: TYPE GRAPH$ENTRY, ! An entry in the Type Mapping Table
6720 6825 2     NAME: REF VECTOR[.BYTE],    ! Name corresponding to typeid
6721 6826 2     SYMID: REF RST$ENTRY,
6722 6827 2     TYPEID: REF RST$ENTRY;      ! Typeid in descriptor
6723 6828 2
6724 6829 2
6725 6830 2 ! For ADA, we map the enumeration type BOOLEAN into the dtype
6726 6831 2 ! DSC$K_DTYPE_TF. The reason for this is that we want the logical
6727 6832 2 ! operators AND, OR, XOR, NOT to work on BOOLEAN (but not other
6728 6833 2 ! enumeration types).
6729 6834 2
6730 6835 2 IF .DBG$GB_LANGUAGE EQL DBG$K_ADA
6731 6836 2 THEN
6732 6837 2 BEGIN
6733 6838 2     IF .LEFT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ENUM
6734 6839 2     THEN
6735 6840 2         BEGIN
6736 6841 2             TYPEID = .LEFT_ARG[DBG$L_DHDR_TYPEID];
6737 6842 2             IF .TYPEID NEQ 0
6738 6843 2             THEN
6739 6844 2                 BEGIN
6740 6845 2                     DSTPTR = .TYPEID[RST$L_DSTPTR];
6741 6846 2                     NAME = DBG$GET_DST_NAME(.DSTPTR);
6742 6847 2                     IF .NAME[0] EQL 7
6743 6848 2                     THEN
6744 6849 2                         BEGIN
6745 6850 2                             IF CH$EQL(7, NAME[1], 7, UPLIT ('BOOLEAN'))
6746 6851 2                             THEN
6747 6852 2                                 BEGIN
6748 6853 2                                     ! We have determined we have data of type 'BOOLEAN'.
6749 6854 2                                     ! Do some additional checking here:
6750 6855 2
```

```

6751 6856 7
6752 6857 7
6753 6858 7
6754 6859 7
6755 6860 7
6756 6861 7
6757 6862 7
6758 6863 8
6759 6864 8
6760 6865 8
6761 6866 8
6762 6867 8
6763 6868 8
6764 6869 8
6765 6870 9
6766 6871 9
6767 6872 9
6768 6873 9
6769 6874 10
6770 6875 10
6771 6876 10
6772 6877 10
6773 6878 10
6774 6879 10
6775 6880 11
6776 6881 11
6777 6882 11
6778 6883 11
6779 6884 12
6780 6885 12
6781 6886 12
6782 6887 12
6783 6888 12
6784 6889 12
6785 6890 12
6786 6891 12
6787 6892 12
6788 6893 11
6789 6894 10
6790 6895 9
6791 6896 8
6792 6897 7
6793 6898 6
6794 6899 5
6795 6900 4
6796 6901 3
6797 6902 2
6798 6903 2
6799 6904 2
6800 6905 2
6801 6906 2
6802 6907 2
6803 6908 2
6804 6909 2
6805 6910 2
6806 6911 2
6807 6912 2

```

```

! check that the enumeration elements are
! (FALSE, TRUE). The reason is that the user may
! have redefined BOOLEAN so we may not be looking
! at a "real" boolean type.
IF .TYPEID[RST$$_TYP$COMPNT] EQL 2
THEN
  BEGIN
    COMPLIST = TYPEID[RST$$_TYP$COMPLST];
    SYMID = .COMPLIST[0];
    DSTPTR = .SYMID[RST$$_DSTPTR];
    NAME = DBG$GET_DST_NAME(.DSTPTR);
    IF .NAME[0] EQL 5
    THEN
      BEGIN
        IF CH$EQL(5, NAME[1],
                  5, UPLIT ($ASCII 'FALSE'))
        THEN
          BEGIN
            SYMID = .COMPLIST[1];
            DSTPTR = .SYMID[RST$$_DSTPTR];
            NAME = DBG$GET_DST_NAME(.DSTPTR);
            IF .NAME[0] EQL 4
            THEN
              BEGIN
                IF CH$EQL(4, NAME[1],
                          4, UPLIT ($ASCII 'TRUE'))
                THEN
                  BEGIN
                    ! We have finally decided that it
                    ! is OK to change the dtype to TF.
                    LEFT_ARG[DBG$$_DHDR FCODE] = RST$$_TYPE ATOMIC;
                    LEFT_ARG[DBG$$_VALUE CLASS] = DSC$$_CLASS S;
                    LEFT_ARG[DBG$$_VALUE DTYPE] = DSC$$_DTYPE TF;
                    LEFT_ARG[DBG$$_VALUE LENGTH] = 1;
                    END;
                  END;
                END;
              END;
            END;
          END;
        END;
      END;
    END;
  END;
END;

! Consult the Type Mapping Table to see if we need to change the dtype
! of either the left or the right operand.
LEFT_TYPE = DBG$GET_DTYPE (.LEFT_ARG);
IF .MAP_TBL NEQ TAB$BASE
THEN
  BEGIN
    INCR I FROM 0 TO .MAP_TBL_SIZE - 1 DO

```

```
BEGIN
MAP_TBL_ENTRY = .MAP_TBL[.I];
IF .MAP_TBL_ENTRY EQL 0 THEN EXITLOOP;
IF .MAP_TBL_ENTRY[TYPE_GRAPH$B_LOWER_TYPE] EQL .LEFT_TYPE
THEN
    BEGIN
        ! We have found a mapping. Change all the information
        ! in the descriptor accordingly.
        LEFT_TYPE = .MAP_TBL_ENTRY[TYPE_GRAPH$B_HIGHER_TYPE];
        ! If we are mapping to a non-vax_standard type, we need
        ! to change the fcode.
        IF .LEFT_TYPE GTR DBG$K_MAXIMUM_DTYPE
        THEN
            BEGIN
                LEFT_ARG(DBG$B_DHDR_FCODE) = .LEFT_TYPE - DBG$K_MAXIMUM_DTYPE;
                LEFT_ARG(DBG$B_VALUE_CLASS) = DSC$R_CLASS_Z;
                LEFT_ARG(DBG$B_VALUE_DTYPE) = DSC$K_DTYPE_Z;
            END
        ELSE
            BEGIN
                LEFT_ARG(DBG$B_DHDR_FCODE) = RST$K_TYPE_ATOMIC;
                LEFT_ARG(DBG$B_VALUE_CLASS) = DBG$MAP_DTYPE_CLASS(
                    .LEFT_TYPE,
                    (IF .LEFT_ARG(DBG$B_VALUE_CLASS) EQL DSC$K_CLASS_SD
                     THEN TRUE ELSE FALSE));
                LEFT_ARG(DBG$B_VALUE_DTYPE) = .LEFT_TYPE;
            END;
        END;
    END;
END;
! End of INCR.
END;
! End of DBG$DO_MAPPING
```

|    |           |    |    |       |    |    |    |       |        |        |                    |      |                        |      |      |
|----|-----------|----|----|-------|----|----|----|-------|--------|--------|--------------------|------|------------------------|------|------|
|    |           |    |    |       |    |    |    |       |        | .PSECT | DBG\$PLIT,NOWRT,   | SHR, | PIC,0                  |      |      |
| 00 | 4E        | 41 | 45 | 4C    | 4F | 4F | 42 | 05C57 |        | .BLKB  | 1                  |      |                        |      |      |
| 00 | 00        | 00 | 45 | 53    | 4C | 41 | 46 | 05C58 | P.ALO: | .ASCII | \BOOLEAN\<0>       | :    |                        |      |      |
|    |           |    |    | 45    | 55 | 52 | 54 | 05C60 | P.ALP: | .ASCII | \FALSE\<0><0><0>   | :    |                        |      |      |
|    |           |    |    |       |    |    |    | 05C68 | P.ALQ: | .ASCII | \TRUE\             | :    |                        |      |      |
|    |           |    |    |       |    |    |    |       |        | .PSECT | DBG\$CODE,NOWRT,   | SHR, | PIC,0                  |      |      |
|    |           |    |    |       |    |    |    |       |        | .ENTRY | DBG\$DO_MAPPING,   | Save | R2,R3,R4,R5,R6,R7,R8,- | :    | 6800 |
|    |           |    |    |       |    |    |    |       |        |        | R9,R10,R11         | :    |                        |      |      |
|    |           |    |    |       |    |    |    |       |        | MOVAB  | MAP_TBL,           | R11  | :                      |      |      |
|    |           |    |    |       |    |    |    |       |        | MOVAB  | DBG\$GET_DST_NAME, | R10  | :                      |      |      |
|    |           |    |    |       |    |    |    |       |        | MOVAB  | P.ALO,             | R9   | :                      |      |      |
| 5B | 00000000' | EF | 9E | 00002 |    |    |    |       |        | CMPB   | DBG\$GB_LANGUAGE,  | #9   | :                      | 6835 |      |
| 5A | 00000000G | 00 | 9E | 00009 |    |    |    |       |        | BNEQ   | 1\$                | :    |                        |      |      |
| 59 | 00000000' | EF | 9E | 00010 |    |    |    |       |        | MOVL   | LEFT_ARG,          | R7   | :                      | 6838 |      |
| 09 | 00000000G | 00 | 91 | 00017 |    |    |    |       |        |        |                    |      |                        |      |      |
|    |           | 75 | 12 | 0001E |    |    |    |       |        |        |                    |      |                        |      |      |
| 57 | 04        | AC | DO | 00020 |    |    |    |       |        |        |                    |      |                        |      |      |

|    |    |       |    |      |    |       |       |                                  |      |
|----|----|-------|----|------|----|-------|-------|----------------------------------|------|
|    |    | 04    | 06 | A7   | 91 | 00024 | CMPB  | 6(R7), #4                        |      |
|    |    | 54    | 08 | 6B   | 12 | 00028 | BNEQ  | 1\$                              | 6841 |
|    |    |       |    | A7   | D0 | 0002A | MOVL  | 8(R7), TYPEID                    | 6842 |
|    |    | 58    | 0C | 65   | 13 | 0002E | BEQL  | 1\$                              | 6845 |
|    |    |       |    | A4   | D0 | 00030 | MOVL  | 12(TYPEID), DSTPTR               | 6846 |
|    |    | 6A    |    | 58   | DD | 00034 | PUSHL | DSTPTR                           |      |
|    |    | 56    |    | 01   | FB | 00036 | CALLS | #1, DBG\$GET_DST_NAME            |      |
|    |    | 07    |    | 50   | D0 | 00039 | MOVL  | R0, NAME                         |      |
|    |    |       |    | 66   | 91 | 0003C | CMPB  | (NAME), #7                       | 6847 |
|    |    |       |    | 54   | 12 | 0003F | BNEQ  | 1\$                              |      |
| 69 | 01 | A6    |    | 07   | 29 | 00041 | CMPC3 | #7, 1(NAME), P.ALQ               | 6850 |
|    |    |       |    | 4D   | 12 | 00046 | BNEQ  | 1\$                              |      |
|    |    | 02    | 28 | A4   | D1 | 00048 | CMPL  | 40(TYPEID), #2                   | 6861 |
|    |    |       |    | 47   | 12 | 0004C | BNEQ  | 1\$                              |      |
|    |    | 54    |    | 2C   | C0 | 0004E | ADDL2 | #44, COMPLIST                    | 6864 |
|    |    | 55    |    | 64   | D0 | 00051 | MOVL  | (COMPLIST), SYMID                | 6865 |
|    |    | 58    | 0C | A5   | D0 | 00054 | MOVL  | 12(SYMID), DSTPTR                | 6866 |
|    |    |       |    | 58   | DD | 00058 | PUSHL | DSTPTR                           | 6867 |
|    |    | 6A    |    | 01   | FB | 0005A | CALLS | #1, DBG\$GET_DST_NAME            |      |
|    |    | 56    |    | 50   | D0 | 0005D | MOVL  | R0, NAME                         |      |
|    |    | 05    |    | 66   | 91 | 00060 | CMPB  | (NAME), #5                       | 6868 |
|    |    |       |    | 30   | 12 | 00063 | BNEQ  | 1\$                              |      |
| 08 | A9 | 01    | A6 | 05   | 29 | 00065 | CMPC3 | #5, 1(NAME), P.ALQ               | 6871 |
|    |    |       |    | 28   | 12 | 00068 | BNEQ  | 1\$                              |      |
|    |    | 55    | 04 | A4   | D0 | 0006D | MOVL  | 4(COMPLIST), SYMID               | 6875 |
|    |    | 58    | 0C | A5   | D0 | 00071 | MOVL  | 12(SYMID), DSTPTR                | 6876 |
|    |    |       |    | 58   | DD | 00075 | PUSHL | DSTPTR                           | 6877 |
|    |    | 6A    |    | 01   | FB | 00077 | CALLS | #1, DBG\$GET_DST_NAME            |      |
|    |    | 56    |    | 50   | D0 | 0007A | MOVL  | R0, NAME                         |      |
|    |    | 04    |    | 66   | 91 | 0007D | CMPB  | (NAME), #4                       | 6878 |
|    |    |       |    | 13   | 12 | 00080 | BNEQ  | 1\$                              |      |
|    |    | 10    | A9 | 01   | A6 | D1    | CMPL  | 1(NAME), P.ALQ                   | 6881 |
|    |    |       |    | 0C   | 12 | 00087 | BNEQ  | 1\$                              |      |
|    |    | 06    | A7 | 02   | 90 | 00089 | MOVB  | #2, 6(R7)                        | 6889 |
|    |    | 14    | A7 | 8F   | D0 | 0008D | MOVL  | #19398657, 20(R7)                | 6892 |
|    |    |       |    | 04   | AC | DD    | PUSHL | LEFT_ARG                         | 6908 |
|    |    | 0000V | CF | 01   | FB | 00098 | CALLS | #1, DBG\$GET_DTYPE               |      |
|    |    |       | 54 | 50   | D0 | 0009D | MOVL  | R0, LEFT_TYPE                    |      |
|    |    |       | 50 | A3A8 | C9 | 9E    | MOVAB | TABLEBASE, R0                    | 6909 |
|    |    |       | 50 |      | 6B | D1    | CMPL  | MAP_TBL, R0                      |      |
|    |    |       |    |      | 5A | 13    | BEQL  | 7\$                              |      |
|    |    | 56    | 04 | AB   | D0 | 000AA | MOVL  | MAP_TBL_SIZE, R6                 | 6912 |
|    |    | 53    |    | 01   | CE | 000AE | MNEGL | #1, -1                           |      |
|    |    |       |    | 4D   | 11 | 000B1 | BRB   | 6\$                              |      |
|    |    | 55    | 00 | BB43 | B0 | 000B3 | MOVW  | @MAP_TBL[1], MAP_TBL_ENTRY       | 6914 |
|    |    |       |    | 4A   | 13 | 000B8 | BEQL  | 7\$                              | 6915 |
| 54 | 55 | 08    |    | 00   | ED | 000BA | CMPTV | #0, #8, MAP_TBL_ENTRY, LEFT_TYPE | 6916 |
|    |    |       |    | 3F   | 12 | 000BF | BNEQ  | 6\$                              |      |
| 54 | 55 | 08    |    | 08   | EF | 000C1 | EXTZV | #8, #8, MAP_TBL_ENTRY, LEFT_TYPE | 6923 |
|    |    | 50    | 04 | AC   | D0 | 000C6 | MOVL  | LEFT_ARG, R0                     | 6931 |
|    |    | 51    | 04 | AC   | D0 | 000CA | MOVL  | LEFT_ARG, R1                     | 6932 |
|    |    | 52    | 14 | A1   | 9E | 000CE | MOVAB | 20(RT), R2                       |      |
|    |    | 2B    |    | 54   | D1 | 000D2 | CMPL  | LEFT_TYPE, #43                   | 6928 |
|    |    |       |    | 0A   | 15 | 000D5 | BLEQ  | 3\$                              |      |
| 06 | A0 | 54    |    | 2B   | 83 | 000D7 | SUBB3 | #43, LEFT_TYPE, 6(R0)            | 6931 |
|    |    |       | 02 | A2   | B4 | 000DC | CLRW  | 2(R2)                            | 6933 |
|    |    |       |    | 1F   | 11 | 000DF | BRB   | 6\$                              | 6928 |

DBGVALOP  
V04-000

C 10  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGVALOP.B32;1

Page 194  
(29)

|    |       |    |    |    |       |      |        |                          |    |      |
|----|-------|----|----|----|-------|------|--------|--------------------------|----|------|
| 06 | A0    |    | 02 | 90 | 000E1 | 3\$: | MOVB   | #2, 6(R0)                | .. | 6937 |
|    | 09    |    | A2 | 91 | 000E5 |      | CMPB   | 3(R2), #9                | .. | 6940 |
|    |       | 03 | 04 | 12 | 000E9 |      | BNEQ   | 4\$                      | .. |      |
|    |       |    | 01 | DD | 000EB |      | PUSHL  | #1                       | .. |      |
|    |       |    | 02 | 11 | 000ED |      | BRB    | 5\$                      | .. |      |
|    |       |    | 7E | D4 | 000EF | 4\$: | CLRL   | -(SP)                    | .. |      |
|    |       |    | 54 | DD | 000F1 | 5\$: | PUSHL  | LEFT, TYPE               | .. | 6939 |
|    | 0000V | CF | 02 | FB | 000F3 |      | CALLS  | #2, DBG\$MAP_DTYPE_CLASS | .. |      |
|    | 03    | A2 | 50 | 90 | 000F8 |      | MOVB   | R0, 3(R2)                | .. |      |
|    | 02    | A2 | 54 | 90 | 000FC |      | MOVB   | LEFT, TYPE, 2(R2)        | .. | 6942 |
| AF |       | 53 | 56 | F2 | 00100 | 6\$: | AOBLSS | R6, T, 2\$               | .. | 6912 |
|    |       |    |    | 04 | 00104 | 7\$: | RET    |                          | .. | 6947 |

; Routine Size: 261 bytes, Routine Base: DBG\$CODE + 0983

```
6844 6948 1 GLOBAL ROUTINE DBG$EVALOP_SET_LANGUAGE (LANGUAGE) : NOVALUE =
6845 6949 1
6846 6950 1 FUNCTION
6847 6951 1     This routine gets called during processing of the SET LANGUAGE
6848 6952 1     command. It sets up the pointer to the Operator Information Table
6849 6953 1     and Type Conversion Information Table for the language being set.
6850 6954 1
6851 6955 1 INPUTS
6852 6956 1     LANGUAGE          - The language code for the language being set
6853 6957 1
6854 6958 1 OUTPUTS
6855 6959 1     NONE
6856 6960 1
6857 6961 1 BEGIN
6858 6962 1
6859 6963 1     ! Case on the language, and set the own variable OPINFO TABLE and
6860 6964 1     ! CVTINFO TABLE to point to the Operator Information Table and Type
6861 6965 1     ! Conversion Table for that language.
6862 6966 1
6863 6967 1 CASE .LANGUAGE FROM DBG$K_MIN_LANGUAGE TO DBG$K_MAX_LANGUAGE OF
6864 6968 1     SET
6865 6969 1     [DBG$K_ADA]:
6866 6970 1         BEGIN
6867 6971 1             OPINFO TABLE = ADA_OPINFO_TABLE;
6868 6972 1             CVTINFO_TABLE = ADA_CVTINFO_TABLE;
6869 6973 1             END;
6870 6974 1
6871 6975 1     [DBG$K_BASIC]:
6872 6976 1         BEGIN
6873 6977 1             OPINFO TABLE = BASIC_OPINFO_TABLE;
6874 6978 1             CVTINFO_TABLE = BASIC_CVTINFO_TABLE;
6875 6979 1             END;
6876 6980 1
6877 6981 1     [DBG$K_BLISS]:
6878 6982 1         BEGIN
6879 6983 1             OPINFO TABLE = BLISS_OPINFO_TABLE;
6880 6984 1             CVTINFO_TABLE = BLISS_CVTINFO_TABLE;
6881 6985 1             END;
6882 6986 1
6883 6987 1     [DBG$K_C]:
6884 6988 1         BEGIN
6885 6989 1             OPINFO TABLE = C_OPINFO_TABLE;
6886 6990 1             CVTINFO_TABLE = C_CVTINFO_TABLE;
6887 6991 1             END;
6888 6992 1
6889 6993 1     [DBG$K_COBOL]:
6890 6994 1         BEGIN
6891 6995 1             OPINFO TABLE = COBOL_OPINFO_TABLE;
6892 6996 1             CVTINFO_TABLE = COBOL_CVTINFO_TABLE;
6893 6997 1             END;
6894 6998 1
6895 6999 1     [DBG$K_FORTRAN]:
6896 7000 1         BEGIN
6897 7001 1             OPINFO TABLE = FORTRAN_OPINFO_TABLE;
6898 7002 1             CVTINFO_TABLE = FORTRAN_CVTINFO_TABLE;
6899 7003 1             END;
6900 7004 1
```

```

6901 7005
6902 7006
6903 7007
6904 7008
6905 7009
6906 7010
6907 7011
6908 7012
6909 7013
6910 7014
6911 7015
6912 7016
6913 7017
6914 7018
6915 7019
6916 7020
6917 7021
6918 7022
6919 7023
6920 7024
6921 7025
6922 7026
6923 7027
6924 7028
6925 7029
6926 7030
6927 7031
6928 7032
6929 7033
6930 7034
6931 7035
6932 7036
6933 7037
6934 7038
6935 7039
6936 7040
6937 7041
6938 7042
6939 7043
6940 7044
6941 7045
6942 7046
6943 7047
6944 7048
6945 7049
6946 7050
6947 7051
6948 7052
6949 7053
6950 7054
6951 7055
6952 7056
6953 7057
6954 7058
6955 7059
6956 7060
6957 7061

```

```

[DBG$K_MACRO]:
BEGIN
  OPINFO_TABLE = MACRO_OPINFO_TABLE;
  CVTINFO_TABLE = MACRO_CVTINFO_TABLE;
END;

```

```

[DBG$K_PASCAL]:
BEGIN
  OPINFO_TABLE = PASCAL_OPINFO_TABLE;
  CVTINFO_TABLE = PASCAL_CVTINFO_TABLE;
END;

```

```

[DBG$K_PLI]:
BEGIN
  OPINFO_TABLE = PLI_OPINFO_TABLE;
  CVTINFO_TABLE = PLI_CVTINFO_TABLE;
END;

```

```

[DBG$K_RPG]:
BEGIN
  OPINFO_TABLE = RPG_OPINFO_TABLE;
  CVTINFO_TABLE = RPG_CVTINFO_TABLE;
END;

```

```

[DBG$K_UNKNOWN]:
BEGIN
  OPINFO_TABLE = UNKNOWN_OPINFO_TABLE;
  CVTINFO_TABLE = UNKNOWN_CVTINFO_TABLE;
END;

```

```

[INRANGE]:
  $DBG_ERROR ('DBGVALOP\DBG$EVALOP_SET_LANGUAGE');

```

```

[OUTRANGE]:
BEGIN
  OPINFO_TABLE = UNKNOWN_OPINFO_TABLE;
  CVTINFO_TABLE = UNKNOWN_CVTINFO_TABLE;
END;

```

TES;

```

! Load in Type Mapping Table from Type Conversion Information Table
! for the current language.

```

```

MAP_TBL = .CVTINFO_TABLE[CVTINFO$M_MAP_TBL] + TABLEBASE;
IF .MAP_TBL NEQ TABLEBASE THEN MAP_TBL_SIZE = .(.MAP_TBL - 4) * 2;

```

```

! Load in Type Conversion Table from Type Conversion Information Table
! for the current language.

```

```

CVT_TBL = .CVTINFO_TABLE[CVTINFO$M_CVT_TBL] + TABLEBASE;
IF .CVT_TBL NEQ TABLEBASE THEN CVT_TBL_SIZE = .(.CVT_TBL - 4) * 1;

```

```

: 6958      7062 2      ! Set up the conversion rounding flag.
: 6959      7063 2
: 6960      7064 2
: 6961      7065 2
: 6962      7066 2
: 6963      7067 1
          CVT_ROUND_FLAG = .CVTINFO_TABLE[CVTINFO$V_ROUND];
          RETURN;
          END;

```

```

.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 21 05C6C P.ALR: .ASCII \!DBG$EVALOP\<92>\DBG$EVALOP_SET_LANGUAGE\
47 4E 41 4C 5F 54 45 53 5F 50 4F 4C 41 56 45 05C7B
          45 47 41 55 05C8A

```

004C  
003F

0A  
0032  
0073  
009A

```

          54 00000000' EF 9E 00000
          53 00000000' EF 9E 00009
          00 04 AC CF 00010
0059 0066 00015 1$:
0080 0025 0001D
0019 008D 00025

```

```

10 A3 0430 0081 31 0002B 2$:
   63 04 A4 9E 0002E
          73 11 00034
10 A3 0D5C C4 9E 0003A 3$:
   63 0870 C4 9E 00040
          73 11 00045
10 A3 13E0 C4 9E 00047 4$:
   63 11A4 C4 9E 0004D
          66 11 00052
10 A3 1CF8 C4 9E 00054 5$:
   63 1820 C4 9E 0005A
          59 11 0005F
10 A3 23A8 C4 9E 00061 6$:
   63 2144 C4 9E 00067
          4C 11 0006C
10 A3 2D8C C4 9E 0006E 7$:
   63 27F4 C4 9E 00074
          3F 11 00079
10 A3 3414 C4 9E 0007B 8$:
   63 31D4 C4 9E 00081
          32 11 00086
10 A3 3D10 C4 9E 00088 9$:

```

```

.PSECT DBG$CODE,NOWRT, SHR, PIC,0
.ENTRY DBG$EVALOP_SET_LANGUAGE, Save R2,R3,R4
MOVAB TABLEBASE, R4
MOVAB CVTINFO_TABLE, R3
CASEL LANGUAGE, #0, #10
.WORD 8$-1$, -
      7$-1$, -
      4$-1$, -
      6$-1$, -
      3$-1$, -
      10$-1$, -
      9$-1$, -
      5$-1$, -
      11$-1$, -
      2$-1$, -
      13$-1$
      13$
BRW 13$
MOVAB ADA_OPINFO_TABLE, OPINFO_TABLE
MOVAB ADA_CVTINFO_TABLE, CVTINFO_TABLE
BRB 12$
MOVAB BASIC_OPINFO_TABLE, OPINFO_TABLE
MOVAB BASIC_CVTINFO_TABLE, CVTINFO_TABLE
BRB 14$
MOVAB BLISS_OPINFO_TABLE, OPINFO_TABLE
MOVAB BLISS_CVTINFO_TABLE, CVTINFO_TABLE
BRB 14$
MOVAB C_OPINFO_TABLE, OPINFO_TABLE
MOVAB C_CVTINFO_TABLE, CVTINFO_TABLE
BRB 14$
MOVAB COBOL_OPINFO_TABLE, OPINFO_TABLE
MOVAB COBOL_CVTINFO_TABLE, CVTINFO_TABLE
BRB 14$
MOVAB FORTRAN_OPINFO_TABLE, OPINFO_TABLE
MOVAB FORTRAN_CVTINFO_TABLE, CVTINFO_TABLE
BRB 14$
MOVAB MACRO_OPINFO_TABLE, OPINFO_TABLE
MOVAB MACRO_CVTINFO_TABLE, CVTINFO_TABLE
BRB 14$
MOVAB PASCAL_OPINFO_TABLE, OPINFO_TABLE

```

```

: 6948
:
: 6968
:
: 7041
: 6972
: 6973
: 6968
: 6978
: 6979
: 6968
: 6984
: 6985
: 6968
: 6990
: 6991
: 6968
: 6996
: 6997
: 6968
: 7002
: 7003
: 6968
: 7008
: 7009
: 6968
: 7014

```

|    |    |    |    |         |    |    |       |       |       |                                      |  |      |
|----|----|----|----|---------|----|----|-------|-------|-------|--------------------------------------|--|------|
|    |    |    | 63 | 385C    | C4 | 9E | 0008E |       | MOVAB | PASCAL_CVTINFO_TABLE, CVTINFO_TABLE  |  | 7015 |
|    |    |    |    |         | 25 | 11 | 00093 |       | BRB   | 14\$                                 |  | 6968 |
|    |    | 10 | A3 | 4778    | C4 | 9E | 00095 | 10\$: | MOVAB | PLI_OPINFO_TABLE, OPINFO_TABLE       |  | 7020 |
|    |    |    | 63 | 4194    | C4 | 9E | 0009B |       | MOVAB | PLI_CVTINFO_TABLE, CVTINFO_TABLE     |  | 7021 |
|    |    |    |    |         | 18 | 11 | 000A0 |       | BRB   | 14\$                                 |  | 6968 |
|    |    | 10 | A3 | 4D54    | C4 | 9E | 000A2 | 11\$: | MOVAB | RPG_OPINFO_TABLE, OPINFO_TABLE       |  | 7026 |
|    |    |    | 63 | 4888    | C4 | 9E | 000A8 |       | MOVAB | RPG_CVTINFO_TABLE, CVTINFO_TABLE     |  | 7027 |
|    |    |    |    |         | 0B | 11 | 000AD | 12\$: | BRB   | 14\$                                 |  | 6968 |
|    |    | 10 | A3 | 5634    | C4 | 9E | 000AF | 13\$: | MOVAB | UNKNOWN_OPINFO_TABLE, OPINFO_TABLE   |  | 7032 |
|    |    |    | 63 | 5194    | C4 | 9E | 000B5 |       | MOVAB | UNKNOWN_CVTINFO_TABLE, CVTINFO_TABLE |  | 7033 |
|    |    |    | 52 |         | 63 | D0 | 000BA | 14\$: | MOVL  | CVTINFO_TABLE, R2                    |  | 7051 |
|    |    |    | 50 |         | 64 | 9E | 000BD |       | MOVAB | TABLEBASE, R0                        |  |      |
| 04 | A3 |    | 50 |         | 62 | C1 | 000C0 |       | ADDL3 | (R2), R0, MAP_TBL                    |  |      |
|    |    |    | 50 | 04      | A3 | D0 | 000C5 |       | MOVL  | MAP_TBL, R0                          |  | 7052 |
|    |    |    | 51 |         | 64 | 9E | 000C9 |       | MOVAB | TABLEBASE, R1                        |  |      |
|    |    |    | 51 |         | 50 | D1 | 000CC |       | CMPL  | R0, R1                               |  |      |
|    |    |    |    |         | 06 | 13 | 000CF |       | BEQL  | 15\$                                 |  |      |
| 08 | A3 | FC | A0 |         | 01 | 78 | 000D1 |       | ASHL  | #1, -4(R0), MAP_TBL_SIZE             |  |      |
|    |    |    | 50 |         | 64 | 9E | 000D7 | 15\$: | MOVAB | TABLEBASE, R0                        |  | 7058 |
|    |    | F8 | A3 | 04 B240 | A3 | D0 | 000DA |       | MOVAB | 24(R2)(R0), CVT_TBL                  |  |      |
|    |    |    | 50 | F8      | A3 | D0 | 000E0 |       | MOVL  | CVT_TBL, R0                          |  | 7059 |
|    |    |    | 51 |         | 64 | 9E | 000E4 |       | MOVAB | TABLEBASE, R1                        |  |      |
|    |    |    | 51 |         | 50 | D1 | 000E7 |       | CMPL  | R0, R1                               |  |      |
|    |    |    |    |         | 05 | 13 | 000EA |       | BEQL  | 16\$                                 |  |      |
| F4 | A3 | 08 | A2 | FC      | A3 | A0 | D0    | 000EC | MOVAB | -4(R0), CVT_TBL_SIZE                 |  |      |
|    |    |    | 01 | FC      | A0 | D0 | 000EC | 16\$: | EXTZV | #0, #1, 8(R2), CVT_ROUND_FLAG        |  | 7064 |
|    |    |    |    |         | 00 | EF | 000F1 |       | RET   |                                      |  | 7067 |
|    |    |    |    |         | 04 | 00 | 000F8 |       |       |                                      |  |      |

; Routine Size: 249 bytes, Routine Base: DBG\$CODE + 0A88

```
6965 7068 1 GLOBAL ROUTINE DBGSEVAL_LANG_OPERATOR (OPERATOR, LEFT_ARG, RIGHT_ARG) =
6966 7069 1
6967 7070 1 FUNCTION
6968 7071 1     This routine does the evaluation of a language expression operator.
6969 7072 1
6970 7073 1     It determines the operand type or types, it determines what type
6971 7074 1     conversions must be done on the operands (if any), it calls the
6972 7075 1     type conversion routines to do whatever conversions are needed, it selects
6973 7076 1     the actual routine to do the operation based on the final operand data
6974 7077 1     types, and it calls the DBG$PERFORM_OPERATOR routine to actually perform
6975 7078 1     the selected operation.
6976 7079 1
6977 7080 1     This routine uses the Operator Routine Table for this operator to
6978 7081 1     determine which actual routine will do the operation on operands of
6979 7082 1     the given data types. If a routine is not found to handle the operation
6980 7083 1     on the given types, then the Type Hierarchy Table
6981 7084 1     is consulted to determine what type conversions
6982 7085 1     may have to be done. If type conversions need to be done,
6983 7086 1     DBG$TYPE CONV is called to actually do the conversions.
6984 7087 1     DBG$PERFORM_OPERATOR is then called with an index specifying the desired
6985 7088 1     operator routine--that is where the actual operation is finally done.
6986 7089 1
6987 7090 1     The routine result Value Descriptor is then returned as the value
6988 7091 1     of the DBGSEVAL_LANG_OPERATOR call.
6989 7092 1
6990 7093 1     A Primary Descriptor may be returned as the routine value as well,
6991 7094 1     if the computation involved is address-valued operands as in C.
6992 7095 1
6993 7096 1 INPUTS
6994 7097 1     OPERATOR - A pointer to the Operator Lexical Token Entry for the
6995 7098 1                 operator to be evaluated.
6996 7099 1
6997 7100 1     LEFT_ARG - A pointer to the left argument Descriptor.
6998 7101 1                 If the operator
6999 7102 1                 is a unary operator, LEFT_ARG points to the operator's one
7000 7103 1                 argument.
7001 7104 1
7002 7105 1     RIGHT_ARG - A pointer to the right argument Descriptor.
7003 7106 1                 If the operator
7004 7107 1                 is a unary operator, RIGHT_ARG must be zero.
7005 7108 1
7006 7109 1 OUTPUTS
7007 7110 1     A pointer to the Value Descriptor which results from the evaluation of
7008 7111 1     the operator is returned as this routine's result. Or,
7009 7112 1     A pointer to the Primary Descriptor.
7010 7113 1
7011 7114 1
7012 7115 2 BEGIN
7013 7116 2
7014 7117 2 MAP
7015 7118 2     OPERATOR: REF TOKEN$ENTRY,      ! Token Entry for operator to perform
7016 7119 2     LEFT_ARG: REF DBG$VALDESC,      ! Left operand Token Entry
7017 7120 2     RIGHT_ARG: REF DBG$VALDESC;      ! Right operand Token Entry
7018 7121 2
7019 7122 2 LOCAL
7020 7123 2     HIER_TBL: REF VECTOR [,WORD],    ! Pointer to a Type Hierarchy Table
7021 7124 2     HIER_TBL_SIZE,                  ! Number of entries in HIER_TBL
```

```
7022 7125 2 INCOMP_TBL: REF VECTOR [,WORD], | Pointer to a Type Incompatibility Table
7023 7126 2 INCOMP_TBL_SIZE, | Number of entries in INCOMP_TBL
7024 7127 2 ROUT_TBL: REF ORT$TABLE, | Pointer to an Operator Routine Table
7025 7128 2 ROUT_TBL_SIZE, | Number of entries in ROUT_TBL
7026 7129 2 BINARY_FLAG: BYTE, | A flag set to TRUE for binary operators
7027 7130 2 DIGITS, | The number of digits for packed decimal
7028 7131 2 LEFT_TYPE, | Dtype for original left operand type
7029 7132 2 LENGTH, | Length of the data
7030 7133 2 MAP_TBL_ENTRY: TYPE GRAPH$ENTRY, | An entry in the Type Mapping Table
7031 7134 2 NEW_LEFT_ARG: REF DBG$VALDESC, | Pointer to left argument Token Entry
7032 7135 2 | after type conversion if needed
7033 7136 2 NEW_LEFT_TYPE, | Dtype for converted left operand type
7034 7137 2 NEW_RIGHT_ARG: REF DBG$VALDESC, | Pointer to right argument Token Entry
7035 7138 2 | after type conversion if needed
7036 7139 2 NEW_RIGHT_TYPE, | Dtype for converted right operand type
7037 7140 2 OPCODE, | Operator code for current operator
7038 7141 2 SD_CLASS_FLAG, | Flag to indicate SD class overrides
7039 7142 2 RESULT: REF DBG$VALDESC, | Pointer to result value descriptor
7040 7143 2 RESULT_TYPE, | Dtype of result of operation
7041 7144 2 RIGHT_TYPE, | Dtype for original right operand type
7042 7145 2 ROUT_TBL_INDEX, | Index into Operator Routine Table
7043 7146 2 ROUT_INDEX, | Routine CASE index for code that does
7044 7147 2 | operation on operand types
7045 7148 2 SYMID: REF RST$ENTRY, | Symid in primary descriptor
7046 7149 2 TYPEID, | Pointer to a typeid
7047 7150 2 TYPEID_INDEX, | Routine CASE index for code that does
7048 7151 2 | typeid check
7049 7152 2 TYPES: TYPE$PAIR, | Type indices for original operand
7050 7153 2 | types in Type Conv Table format
7051 7154 2
7052 7155 2
7053 7156 2
7054 7157 2
7055 7158 2
7056 7159 2
7057 7160 2
7058 7161 2
7059 7162 2
7060 7163 2
7061 7164 2
7062 7165 2
7063 7166 2
7064 7167 2
7065 7168 2
7066 7169 2
7067 7170 2
7068 7171 2
7069 7172 2
7070 7173 2
7071 7174 2
7072 7175 2
7073 7176 2
7074 7177 2
7075 7178 2
7076 7179 2
7077 7180 2
7078 7181 2

! Trap out one special case. If the incoming token is Unary minus or
! Unary plus, and the unconverted bit is on, we know we have a
! constant, say like -1. We do not want to turn this -1 into negative
! integer right away, for those languages have packed decimal we are
! not sure at this point we have an integer or packed decimal number.
! We make this decision at the evaluation time or depositing time.
! So if we have one of this token, we mark it, then simply returns.
IF .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_UNARY_PLUS OR
.OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_UNARY_MINUS
THEN
  BEGIN
    IF .LEFT_ARG[DBG$V_DHDR_UNCVT]
    THEN
      BEGIN
        ! In this case, there should not be any right argument.
        !
        IF .RIGHT_ARG NEQ 0
        THEN
          SDBG_ERROR('DBGEVALOP\DBG$EVAL_LANG_OPERATOR, inconsistency 1');

        ! In this case, the left arg. should not be primary or volatile
        ! value descriptor.
```

```
7079 7182 4
7080 7183 4
7081 7184 5
7082 7185 4
7083 7186 4
7084 7187 4
7085 7188 4
7086 7189 4
7087 7190 4
7088 7191 4
7089 7192 4
7090 7193 4
7091 7194 4
7092 7195 5
7093 7196 5
7094 7197 5
7095 7198 5
7096 7199 5
7097 7200 5
7098 7201 5
7099 7202 5
7100 7203 4
7101 7204 4
7102 7205 4
7103 7206 5
7104 7207 5
7105 7208 5
7106 7209 5
7107 7210 5
7108 7211 5
7109 7212 5
7110 7213 5
7111 7214 4
7112 7215 4
7113 7216 4
7114 7217 4
7115 7218 5
7116 7219 5
7117 7220 5
7118 7221 5
7119 7222 5
7120 7223 5
7121 7224 5
7122 7225 5
7123 7226 5
7124 7227 5
7125 7228 5
7126 7229 5
7127 7230 5
7128 7231 5
7129 7232 5
7130 7233 5
7131 7234 5
7132 7235 5
7133 7236 5
7134 7237 5
7135 7238 5

!
IF (.LEFT_ARG[DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC) OR
(.LEFT_ARG[DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC)
THEN
    $DBG_ERROR('DBGEVALOP\DBG$EVAL_LANG_OPERATOR, inconsistency 2');

! Mark the sign and return. Check to see if it is already marked,
! if it is, flip the sign, for we may have -(-1) case, this is
! the same as +1.
IF .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_UNARY_MINUS
THEN
    BEGIN
    IF .LEFT_ARG[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST
    THEN
        LEFT_ARG[DBG$W_VALUE_SIGN_CODE] = TOKEN$K_POSCONST
    ELSE
        LEFT_ARG[DBG$W_VALUE_SIGN_CODE] = TOKEN$K_NEGCONST;
    END
ELSE
    IF .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_UNARY_PLUS
    THEN
        BEGIN
        IF .LEFT_ARG[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST
        THEN
            LEFT_ARG[DBG$W_VALUE_SIGN_CODE] = TOKEN$K_NEGCONST
        ELSE
            LEFT_ARG[DBG$W_VALUE_SIGN_CODE] = TOKEN$K_POSCONST;
        END
    ELSE
        $DBG_ERROR('DBGEVALOP\DBG$EVAL_LANG_OPERATOR, inconsistency 3');

RETURN .LEFT_ARG;
END;

! This is called from DBG$EVAL_LANG_OPEARTOR with TOKEN$K_NEGCONST or
! TOKEN$K_POSCONST token, replaced it to TOKEN$K_NEG_SIGN or
! TOKEN$K_POS_SIGN, so the normal unary minus or unary plus operation
! can take place.
IF .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_NEGCONST
THEN
    OPERATOR = DBG$GL_NEG_SIGN_TOKEN;

IF .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_POSCONST
THEN
    OPERATOR = DBG$GL_POS_SIGN_TOKEN;

! Initialize table pointers and flags.
MAX_DEPTH = 1;
```

```
IF .BLISS_BITSELECTION_FLAG1
THEN
    BLISS_BITSELECTION_FLAG2 = TRUE
ELSE
    BLISS_BITSELECTION_FLAG2 = FALSE;
BLISS_BITSELECTION_FLAG1 = FALSE;
BLISS_INDIRECTION_FLAG = FALSE;
OPCODE = .OPERATOR[TOKEN$W_CODE];
DBG$GL_OPCODE_NAME = OPERATOR[TOKEN$B_OPLEN];
BINARY_FLAG = .OPERATOR[TOKEN$B_KIND] EQL TOKEN$K_INFIX_OP;
ROUT_TBL = .OPINFO_TABLE[OPCODE, OPINFO$ROUT_TBL] + TABLEBASE;
IF .ROUT_TBL EQL TABLEBASE
THEN
    BEGIN
        IF .OPCODE NEQ TOKEN$K_DEPOSIT AND .OPCODE NEQ TOKEN$K_CONVERT
        AND .OPCODE NEQ TOKEN$K_IDENTITY
        THEN
            $DBG_ERROR ('DBGEVALOP\DBG$EVAL_LANG_OPERATOR routine table missing');
        END
    ELSE
        ROUT_TBL_SIZE = (.ROUT_TBL - 4) / 2;
        HIER_TBL = .OPINFO_TABLE[OPCODE, OPINFO$HIER_TBL] + TABLEBASE;
        IF .HIER_TBL NEQ TABLEBASE
        THEN
            BEGIN
                HIER_TBL_SIZE = (.HIER_TBL - 4) * 2;
                MAX_DEPTH = .MAX_DEPTH + .HIER_TBL_SIZE * 2;
            END;
        INCOMP_TBL = .OPINFO_TABLE[OPCODE, OPINFO$INCOMP_TBL] + TABLEBASE;
        IF .INCOMP_TBL NEQ TABLEBASE THEN INCOMP_TBL_SIZE = (.INCOMP_TBL - 4) * 2;

        ! Convert primary descriptors or volatile value descriptors
        ! to value descriptors. Something should be done to
        ! clean up the structure of the code below. The basic idea of what
        ! it is doing is:
        ! If we get a Primary for 'x', say, or a volatile value descriptor for
        ! address '200' in the user program, then:
        ! - If the language does an implicit fetch on operands (i.e., all languages
        !   except BLISS and MACRO), then we do the fetch here by calling PRIM_TO_VAL.
        ! - For BLISS and MACRO we call PRIM_TO_ADDR in order to get the address
        !   of the operand into a value descriptor.
        ! - Special case for arrays in C.
        ! - Special case for records in COBOL.
        ! (It's these special cases that make the code such a mess.)

        IF (.LEFT_ARG[DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC) OR
        (.LEFT_ARG[DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC)
        THEN
            ! Most operators in most languages do an implicit fetch on primaries.
            ! E.g., in FORTRAN, if we say EV X we want to fetch the value at X.
            ! DBG$PRIM_TO_VAL is the routine that converts primaries to value
            ! descriptors, and does the fetch in the process.
            ! The exceptions are: in BLISS, we only do the fetch when we see
            ! the fetch operator (.). Also, for the "address of" operator
            ! (& in C), we want to suppress this fetch. For these cases,
```

```
7193 7296 2
7194 7297 2
7195 7298 2
7196 7299 2
7197 7300 2
7198 7301 2
7199 7302 2
7200 7303 2
7201 7304 2
7202 7305 2
7203 7306 2
7204 7307 2
7205 7308 2
7206 7309 2
7207 7310 2
7208 7311 2
7209 7312 4
7210 7313 4
7211 7314 3
7212 7315 4
7213 7316 4
7214 7317 4
7215 7318 4
7216 7319 4
7217 7320 4
7218 7321 4
7219 7322 4
7220 7323 4
7221 7324 4
7222 7325 4
7223 7326 4
7224 7327 4
7225 7328 4
7226 7329 4
7227 7330 4
7228 7331 4
7229 7332 4
7230 7333 3
7231 7334 4
7232 7335 4
7233 7336 4
7234 7337 4
7235 7338 4
7236 7339 4
7237 7340 4
7238 7341 4
7239 7342 4
7240 7343 4
7241 7344 4
7242 7345 4
7243 7346 4
7244 7347 4
7245 7348 4
7246 7349 4
7247 7350 4
7248 7351 4
7249 7352 5
```

```
! we use a routine DBG$PRIM_TO_ADDR which converts a primary
! descriptor to a value descriptor containing the address of the
! primary.
! We consult a flag in the Operator Information Table that tells
! us whether or not to do this fetch.
IF .OPINFO_TABLE [.OPCODE, OPINFO$V_FETCH]
THEN
  BEGIN
    BLISS_INDIRECTION_FLAG = TRUE;

    ! Special case for arrays in C. We want to treat un-subscripted
    ! arrays as pointers, and be able to add, subtract, and do
    ! dereferencing on the addresses.
    IF (.DBG$GB_LANGUAGE EQL DBG$K_C)
    AND (.LEFT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ARRAY)
    THEN
      BEGIN
        ! Save the typeid.
        TYPEID = .LEFT_ARG[DBG$L_DHDR_TYPEID];

        ! Build a descriptor with the address of the array.
        IF NOT DBG$PRIM_TO_ADDR(.LEFT_ARG, DSC$K_DTYPE_L, LEFT_ARG)
        THEN
          $DBG_ERROR('DBGEVALOP\DBGEVAL_LANG_OPERATOR, prim to addr failed');

        ! Dummy in the FCODE as TPTR. Also fill in the typeid.
        LEFT_ARG[DBG$B_DHDR_FCODE] = RST$K_TYPE_TPTR;
        LEFT_ARG[DBG$L_DHDR_TYPEID] = .TYPEID;
      END
    ELSE
      BEGIN
        ! If we have an aggregate, first try calling the
        ! routine that turns a 1-dimensional array of characters
        ! into a string.
        IF .LEFT_ARG[DBG$V_DHDR_AGGR]
        THEN
          DBG$COLLECT(.LEFT_ARG);

        ! If we still have an aggregate, we check for the special
        ! case of COBOL records. For all other cases, we signal
        ! an error, because operations on aggregates are not
        ! supported.
        SYMID = 0;
        IF .LEFT_ARG[DBG$V_DHDR_AGGR]
        THEN
          BEGIN
```

```
7250 7353 5 IF .LEFT_ARG[DBG$B_DHDR_FCODE] NEQ RST$K_TYPE_RECORD
7251 7354 THEN
7252 7355 SIGNAL(DBG$_NOVALUE);
7253 7356
7254 7357
7255 7358 ! This is record aggregate, if the language code is cobol, treated as text string.
7256 7359
7257 7360 SYMID = .LEFT_ARG[DBG$L_DHDR_SYMID];
7258 7361 IF .SYMID EQL 0 THEN SIGNAL(DBG$_NOVALUE);
7259 7362 WHILE .SYMID[RST$B_KIND] NEQ RST$K_MODULE DO
7260 7363 BEGIN
7261 7364 SYMID = .SYMID[RST$L_UPSCOPEPTR];
7262 7365 IF .SYMID EQL 0 THEN SIGNAL(DBG$_NOVALUE);
7263 7366 END;
7264 7367 IF .SYMID[RST$B_LANGUAGE] NEQ DBG$K_COBOL
7265 7368 THEN
7266 7369 SIGNAL(DBG$_NOVALUE);
7267 7370 END;
7268 7371
7269 7372 IF NOT DBG$PRIM_TO_VAL(.LEFT_ARG, DBG$K_VALUE_DESC, LEFT_ARG)
7270 7373 THEN
7271 7374 $DBG_ERROR('DBGEVALOP\DBG$EVAL_LANG_OPERATOR, prim to val failed');
7272 7375
7273 7376 END;
7274 7377
7275 7378 END
7276 7379
7277 7380 ELSE
7278 7381 IF NOT DBG$PRIM_TO_ADDR(.LEFT_ARG, DSC$K_DTYPE_L, LEFT_ARG)
7279 7382 THEN
7280 7383 $DBG_ERROR('DBGEVALOP\DBG$EVAL_LANG_OPERATOR, prim to addr failed');
7281 7384
7282 7385 ! If this is a binary operator, do the same for the right operand.
7283 7386
7284 7387 IF .BINARY_FLAG
7285 7388 THEN
7286 7389 IF (.RIGHT_ARG[DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC) OR
7287 7390 ((.RIGHT_ARG[DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC) AND
7288 7391 (.OPCODE NEQ TOKEN$K_DEPOSIT) AND (.OPCODE NEQ TOKEN$K_CONVERT))
7289 7392 THEN
7290 7393 IF .OPINFO_TABLE [.OPCODE, OPINFO$V_FETCH]
7291 7394 THEN
7292 7395 BEGIN
7293 7396 BLISS_INDIRECTION_FLAG = TRUE;
7294 7397
7295 7398 ! Special case for arrays in C. We want to treat un-subscripted
7296 7399 ! arrays as pointers, and be able to add, subtract, and do
7297 7400 ! dereferencing on the addresses.
7298 7401
7299 7402 IF (.DBG$GB_LANGUAGE EQL DBG$K_C) AND
7300 7403 (.RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ARRAY)
7301 7404 THEN
7302 7405 BEGIN
7303 7406
7304 7407 ! Save the typeid.
7305 7408
7306 7409 TYPEID = .RIGHT_ARG[DBG$L_DHDR_TYPEID];
```

```
7307 7410 4
7308 7411 4
7309 7412 4
7310 7413 4
7311 7414 4
7312 7415 4
7313 7416 4
7314 7417 4
7315 7418 4
7316 7419 4
7317 7420 4
7318 7421 4
7319 7422 4
7320 7423 3
7321 7424 4
7322 7425 4
7323 7426 4
7324 7427 4
7325 7428 4
7326 7429 4
7327 7430 4
7328 7431 4
7329 7432 4
7330 7433 4
7331 7434 4
7332 7435 4
7333 7436 4
7334 7437 4
7335 7438 4
7336 7439 4
7337 7440 4
7338 7441 4
7339 7442 4
7340 7443 5
7341 7444 5
7342 7445 5
7343 7446 5
7344 7447 5
7345 7448 5
7346 7449 5
7347 7450 5
7348 7451 5
7349 7452 5
7350 7453 5
7351 7454 6
7352 7455 6
7353 7456 6
7354 7457 5
7355 7458 5
7356 7459 5
7357 7460 5
7358 7461 4
7359 7462 4
7360 7463 4
7361 7464 4
7362 7465 4
7363 7466 4
```

```
! Build a descriptor with the address of the array.
! IF NOT DBG$PRIM_TO_ADDR(.RIGHT_ARG, DSC$K_DTYPE_L, RIGHT_ARG)
! THEN
!     $DBG_ERROR('DBGEVALOP\DBG$EVAL_LANG_OPERATOR, prim to addr failed');
! Dummy in the FCODE as TPTR. Also fill in the typeid.
! RIGHT_ARG[DBG$B_DHDR_FCODE] = RST$K_TYPE_TPTR;
! RIGHT_ARG[DBG$L_DHDR_TYPEID] = .TYPEID;
! END
ELSE
BEGIN

! If we have an aggregate, first try calling the
! routine that turns a 1-dimensional array of characters
! into a string.
! IF .RIGHT_ARG[DBG$V_DHDR_AGGR]
! THEN
!     DBG$COLLECT(.RIGHT_ARG);

! If we still have an aggregate, we check for the special
! case of COBOL records. For all other cases, we signal
! an error, because operations on aggregates are not
! supported.
SYMID = 0;
IF .RIGHT_ARG[DBG$V_DHDR_AGGR]
THEN
BEGIN
IF .RIGHT_ARG[DBG$B_DHDR_FCODE] NEQ RST$K_TYPE_RECORD
THEN
    SIGNAL(DBG$_NOVALUE);

! This is record aggregate, if the language code is cobol, treated as text string.
! SYMID = .RIGHT_ARG[DBG$L_DHDR_SYMID0];
! IF .SYMID EQL 0 THEN SIGNAL(DBG$_NOVALUE);
! WHILE .SYMID[RST$B_KIND] NEQ RST$K_MODULE DO
! BEGIN
!     SYMID = .SYMID[RST$L_UPSCOPEPTR];
!     IF .SYMID EQL 0 THEN SIGNAL(DBG$_NOVALUE);
! END;
! IF .SYMID[RST$B_LANGUAGE] NEQ DBG$K_COBOL
! THEN
!     SIGNAL(DBG$_NOVALUE);
! END;

IF NOT DBG$PRIM_TO_VAL(.RIGHT_ARG, DBG$K_VALUE_DESC, RIGHT_ARG)
THEN
    $DBG_ERROR('DBGEVALOP\DBG$EVAL_LANG_OPERATOR, prim to val failed');
```

```
7364 7467 3
7365 7468 3
7366 7469 3
7367 7470 3
7368 7471 3
7369 7472 3
7370 7473 3
7371 7474 3
7372 7475 3
7373 7476 3
7374 7477 3
7375 7478 3
7376 7479 3
7377 7480 3
7378 7481 3
7379 7482 4
7380 7483 4
7381 7484 4
7382 7485 5
7383 7486 5
7384 7487 5
7385 7488 4
7386 7489 3
7387 7490 3
7388 7491 3
7389 7492 3
7390 7493 3
7391 7494 3
7392 7495 3
7393 7496 3
7394 7497 3
7395 7498 3
7396 7499 3
7397 7500 3
7398 7501 3
7399 7502 3
7400 7503 3
7401 7504 3
7402 7505 4
7403 7506 5
7404 7507 5
7405 7508 4
7406 7509 4
7407 7510 4
7408 7511 4
7409 7512 3
7410 7513 3
7411 7514 3
7412 7515 3
7413 7516 3
7414 7517 3
7415 7518 3
7416 7519 3
7417 7520 3
7418 7521 3
7419 7522 3
7420 7523 2

      END;
    ELSE
      IF NOT DBG$PRIM_TO_ADDR(.RIGHT_ARG, DSC$K_DTYPE_L, RIGHT_ARG)
      THEN
        SDBG_ERROR('DBGEVALOP\DBGEVAL_LANG_OPERATOR, prim to addr failed');

! Take care of DEP rfa data type = constant case.
!
IF .OPCODE EQL TOKEN$K_DEPOSIT
THEN
  BEGIN
    IF .LEFT_ARG[DBG$V_DHDR_UNCVT]
    THEN
      BEGIN
        IF .RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_RFA
        THEN
          BEGIN
            LEFT_ARG[DBG$L_DHDR_TYPEID] = .RIGHT_ARG[DBG$L_DHDR_TYPEID];
            LEFT_ARG = DBG$CONV_TRFA_VALUE(.LEFT_ARG);
          END;
        END;
      END;
    END;

! At this point we should have value descriptors. Perform special
! NRO Data type map check for RPG.
!
IF .DBG$GB_LANGUAGE EQL DBG$K_RPG
THEN
  BEGIN
    IF .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_NRO
    THEN
      LEFT_ARG = MAP_NRO_DTYPE_IN_RPG(.LEFT_ARG);

    IF .BINARY_FLAG
    THEN
      BEGIN
        IF (.OPCODE NEQ TOKEN$K_DEPOSIT AND
            .OPCODE NEQ TOKEN$K_CONVERT)
        THEN
          IF .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_NRO
          THEN
            RIGHT_ARG = MAP_NRO_DTYPE_IN_RPG(.RIGHT_ARG);
          END;
        END;
      END;
    END;

! Take care of the [] (empty set).
!
IF .LEFT_ARG[DBG$B_DHDR_LANG] EQL 'X'FF' AND
    .LEFT_ARG[DBG$B_VALUE_CLASS] EQL 'X'FF' AND
    .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL 'X'FF'
THEN
  LEFT_ARG = FIXUP_EMPTY_SET(.LEFT_ARG, .RIGHT_ARG);
```

```
7421 7524 2 IF .BINARY_FLAG
7422 7525 2 THEN
7423 7526 2   IF .RIGHT_ARG[DBG$B_DHDR_LANG] EQL 'X'FF' AND
7424 7527 2   .RIGHT_ARG[DBG$B_VALUE_CLASS] EQL 'X'FF' AND
7425 7528 2   .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL 'X'FF'
7426 7529 2   THEN
7427 7530 2     RIGHT_ARG = FIXUP_EMPTY_SET(.RIGHT_ARG, .LEFT_ARG);
7428 7531 2
7429 7532 2
7430 7533 2   ! Perform 'type mapping'. Do not map left-hand-side of deposit when
7431 7534 2   ! override applied.
7432 7535 2
7433 7536 2   DBG$DO_MAPPING(.LEFT_ARG);
7434 7537 2   IF .BINARY_FLAG
7435 7538 2   THEN
7436 7539 2     IF (.OPCODE NEQ TOKEN$K_DEPOSIT) OR
7437 7540 2     (NOT .RIGHT_ARG[DBG$V_DHDR_OVERRIDE])
7438 7541 2     THEN
7439 7542 2       DBG$DO_MAPPING(.RIGHT_ARG);
7440 7543 2
7441 7544 2
7442 7545 2   ! Get a Type for the left operand.
7443 7546 2   !
7444 7547 2   LEFT_TYPE = DBG$GET_DTYPE(.LEFT_ARG);
7445 7548 2   NEW_LEFT_TYPE = .LEFT_TYPE;
7446 7549 2
7447 7550 2
7448 7551 2   ! Get a Type for the right operand.
7449 7552 2   !
7450 7553 2   IF .BINARY_FLAG THEN
7451 7554 2     BEGIN
7452 7555 2       RIGHT_TYPE = DBG$GET_DTYPE(.RIGHT_ARG);
7453 7556 2       NEW_RIGHT_TYPE = .RIGHT_TYPE;
7454 7557 2     END
7455 7558 2
7456 7559 2   ! Unary operator.
7457 7560 2   !
7458 7561 2   ELSE
7459 7562 2     BEGIN
7460 7563 2       RIGHT_TYPE = 0;
7461 7564 2       NEW_RIGHT_TYPE = 0;
7462 7565 2     END;
7463 7566 2
7464 7567 2
7465 7568 2   ! If we are processing the identity operator, then just return
7466 7569 2   ! the left argument here. Before we return, check to see if any
7467 7570 2   ! unconverted is left undone. Make sure we take care of the
7468 7571 2   ! unary minus/unary plus case that was returned from earlier
7469 7572 2   ! call. (Note: if you EV -1, -1 was not converted the first
7470 7573 2   ! time around, now we are hit by this token$K_identity and
7471 7574 2   ! unconverted constant).
7472 7575 2
7473 7576 2   IF .OPCODE EQL TOKEN$K_IDENTITY
7474 7577 2   THEN
7475 7578 2     BEGIN
7476 7579 2       IF .LEFT_ARG[DBG$V_DHDR_UNCVT]
7477 7580 2       THEN
```

```
7478 7581
7479 7582
7480 7583
7481 7584
7482 7585
7483 7586
7484 7587
7485 7588
7486 7589
7487 7590
7488 7591
7489 7592
7490 7593
7491 7594
7492 7595
7493 7596
7494 7597
7495 7598
7496 7599
7497 7600
7498 7601
7499 7602
7500 7603
7501 7604
7502 7605
7503 7606
7504 7607
7505 7608
7506 7609
7507 7610
7508 7611
7509 7612
7510 7613
7511 7614
7512 7615
7513 7616
7514 7617
7515 7618
7516 7619
7517 7620
7518 7621
7519 7622
7520 7623
7521 7624
7522 7625
7523 7626
7524 7627
7525 7628
7526 7629
7527 7630
7528 7631
7529 7632
7530 7633
7531 7634
7532 7635
7533 7636
7534 7637
```

```
LEFT_ARG = DBG$CONV_TEXT_VALUE(.LEFT_ARG, .LEFT_ARG, 0);
```

```
RETURN .LEFT_ARG;
END;
```

Consult the Type Hierarchy Table to see if any data type conversions need to be done.

There are basically three cases here. They all use the hierarchy table, but in different ways. The FIND\_PATH and FIND\_JOIN routines also use the Operator Routine Table.

1. Binary operators, such as +, which may do implicit conversions on their operands. Here we call the FIND\_JOIN routine. This routine assumes we are trying to convert both left and right operand to the same type. It finds a type that both can be converted to and which is legal for the given operator. If there are several such types, it chooses the one with the shortest conversion path.

2. Unary operators. Here we call the FIND\_PATH routine. This tries to find a type that the operand can be converted to which is legal for the given operator. If there are several, it chooses the one with the shortest conversion path.

3. The ASSIGNMENT operator and the CONVERT operator. Here we know the target type, and we are just trying to determine if there is a path from the source type. We use the routine FIND\_PATH\_DEPOSIT to determine this.

```
IF .BINARY_FLAG
THEN
  BEGIN
```

```
! If an override was present on the DEPOSIT command, e.g.,
DEPOSIT/FLOAT 1000 = 1.1
then we skip the FIND_PATH_DEPOSIT routine which checks whether
the deposit is legal in the current language. We just call
the TYPE_CONV_DEBUG routine directly to do the deposit in a
language-independent fashion. Also call the routine directly
for instruction deposits.
```

```
IF .OPCODE EQL TOKEN$K_DEPOSIT
AND (.RIGHT_ARG [DBG$V_DHDR_OVERRIDE]
OR (.LEFT_TYPE EQL DSC$K_DTYPE_T AND .RIGHT_TYPE EQL DSC$K_DTYPE_ZI))
THEN
```

```
  BEGIN
    IF .LEFT_ARG[DBG$V_DHDR_UNCVT]
    THEN
      LEFT_ARG = DBG$CONV_TEXT_VALUE(.LEFT_ARG, .LEFT_ARG, .RIGHT_TYPE);

    RETURN DBG$COVER_DX_DX (.LEFT_ARG, .RIGHT_ARG, .CVT_ROUND_FLAG);
  END;
```

! Check for the special data, FCODE = PICT, DTYPE = T. for DEPOSIT.

7535 7638 3  
7536 7639 3  
7537 7640 3  
7538 7641 3  
7539 7642 3  
7540 7643 3  
7541 7644 3  
7542 7645 4  
7543 7646 4  
7544 7647 4  
7545 7648 4  
7546 7649 4  
7547 7650 4  
7548 7651 3  
7549 7652 3  
7550 7653 3  
7551 7654 3  
7552 7655 3  
7553 7656 3  
7554 7657 3  
7555 7658 3  
7556 7659 3  
7557 7660 3  
7558 7661 4  
7559 7662 4  
7560 7663 4  
7561 7664 4  
7562 7665 4  
7563 7666 4  
7564 7667 4  
7565 7668 4  
7566 7669 4  
7567 7670 4  
7568 7671 4  
7569 7672 4  
7570 7673 4  
7571 7674 4  
7572 7675 4  
7573 7676 4  
7574 7677 5  
7575 7678 4  
7576 7679 4  
7577 7680 4  
7578 7681 4  
7579 7682 4  
7580 7683 4  
7581 7684 4  
7582 7685 4  
7583 7686 4  
7584 7687 4  
7585 7688 5  
7586 7689 5  
7587 7690 5  
7588 7691 5  
7589 7692 5  
7590 7693 5  
7591 7694 5

```
! In Type Hierd. table, this case is caught by DBG$K_DTYPE_PICT.  
! If both operands are text data, then we want to treat this as  
! standard Text String.  
IF .OPCODE EQL TOKEN$K_DEPOSIT OR  
  .OPCODE EQL TOKEN$K_CONVERT  
THEN  
  BEGIN  
    IF .RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT AND  
      .LEFT_ARG[DBG$B_VALUE_DTYPE] NEQ DSC$K_DTYPE_T AND  
      .DBG$B_LANGUAGE EQL DBG$K_COBOL  
    THEN  
      NEW_RIGHT_TYPE = DBG$K_DTYPE_PICT;  
    END;  
  
IF .OPCODE EQL TOKEN$K_DEPOSIT OR .OPCODE EQL TOKEN$K_CONVERT  
THEN  
  IF FIND PATH DEPOSIT (.NEW_LEFT_TYPE, .NEW_RIGHT_TYPE, 0,  
    .HIER_TBL, .HIER_TBL_SIZE,  
    .INCOMP_TBL, .INCOMP_TBL_SIZE,  
    .ROUT_TBL, .ROUT_TBL_SIZE)  
  THEN  
    BEGIN  
  
      ! Change the dtype for date type PICT, so the LANGUAGE type  
      ! conversion can take place.  
  
      IF .RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT AND  
        .LEFT_ARG[DBG$B_VALUE_DTYPE] NEQ DSC$K_DTYPE_T AND  
        .DBG$B_LANGUAGE EQL DBG$K_COBOL  
      THEN  
        RIGHT_ARG[DBG$B_VALUE_DTYPE] = DBG$K_DTYPE_PICT;  
  
      ! If the target is SD, make the new type to be packed.  
      IF (.RIGHT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD) AND  
        (NOT .RIGHT_ARG[DBG$B_VALUE_FL_BINSCALE])  
      THEN  
        NEW_RIGHT_TYPE = DSC$K_DTYPE_P;  
  
      ! Check for unconverted bit. At this point, long interger  
      ! string or floating string may be converted to Packed  
      ! decimal if the new_right_type is a Packed decimal data item.  
  
      IF .LEFT_ARG[DBG$B_DHDR_UNCVT]  
      THEN  
        BEGIN  
  
          ! This is a special case in COBOL. We can only take the value  
          ! and deposit it into unsigned variable.  
          ! For example: DEP (4V4 (SD, WU) = -1234.1234. The value  
          ! in (4V4 is 1234.1234.
```

7592 7695 5  
7593 7696 5  
7594 7697 5  
7595 7698 5  
7596 7699 6  
7597 7700 6  
7598 7701 6  
7599 7702 6  
7600 7703 6  
7601 7704 6  
7602 7705 6  
7603 7706 6  
7604 7707 6  
7605 7708 6  
7606 7709 7  
7607 7710 7  
7608 7711 7  
7609 7712 8  
7610 7713 8  
7611 7714 8  
7612 7715 7  
7613 7716 7  
7614 7717 6  
7615 7718 6  
7616 7719 5  
7617 7720 5  
7618 7721 5  
7619 7722 5  
7620 7723 4  
7621 7724 4  
7622 7725 4  
7623 7726 4  
7624 7727 4  
7625 7728 4  
7626 7729 4  
7627 7730 5  
7628 7731 4  
7629 7732 4  
7630 7733 4  
7631 7734 5  
7632 7735 4  
7633 7736 4  
7634 7737 4  
7635 7738 4  
7636 7739 4  
7637 7740 4  
7638 7741 4  
7639 7742 4  
7640 7743 4  
7641 7744 4  
7642 7745 4  
7643 7746 4  
7644 7747 4  
7645 7748 4  
7646 7749 4  
7647 7750 4  
7648 7751 4

```
IF .DBG$GB_LANGUAGE EQL DBG$K_COBOL OR
DBG$GB_LANGUAGE EQL DBG$K_RPG
THEN
  BEGIN
    ! Check to see if the target is unsigned data type.
    IF .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_WU OR
      .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_LU OR
      .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_QU OR
      .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_NU
    THEN
      BEGIN
        IF .LEFT_ARG[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST
        THEN
          BEGIN
            LEFT_ARG[DBG$W_VALUE_SIGN_CODE] = TOKEN$K_POSCONST;
            SIGNAL(DBG$_IVALOUTBRDS, T, OPERATOR[TOKEN$B_OPLEN]);
          END;
        END;
      END;
    END;
    LEFT_ARG = DBG$CONV_TEXT_VALUE(.LEFT_ARG, .LEFT_ARG,
                                  .RIGHT_TYPE);
    END;

    ! There are cases the intermediate data type is needed
    ! before the deposit, for example, DEP P=F, F needs to
    ! convert to Packed decimal before deposit into P.
    IF ((.RIGHT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD) AND
        (NOT .RIGHT_ARG[DBG$V_VALUE_FL_BINSCALE])) OR
        (.RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_P) OR
        (.RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_NU) OR
        (.RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DBG$K_DTYPE_PICT)
    THEN
      LEFT_ARG = INTMED_DATA_FOR_DEP(.LEFT_ARG, DSC$K_DTYPE_P,
                                     FALSE);

    ! We catch none-constant cases in here, for the ease of checking
    ! sign for different data types. In cobol, we should have
    ! the source converted into Packed decimal at this point.
    ! (in intermediate).

    ! This is a special case in COBOL. We can only take the value
    ! and deposit it into unsigned variable.
    ! For example: DEP (C4V4 (SD, WU) = SC4V4 (SD, W) The value
    ! in C4V4 is 1234.1234, in SC4V4 is -1234.1234.
    IF .DBG$GB_LANGUAGE EQL DBG$K_COBOL OR
      .DBG$GB_LANGUAGE EQL DBG$K_RPG
```

7649 7752 4  
7650 7753 5  
7651 7754 5  
7652 7755 5  
7653 7756 5  
7654 7757 5  
7655 7758 6  
7656 7759 6  
7657 7760 6  
7658 7761 5  
7659 7762 6  
7660 7763 5  
7661 7764 6  
7662 7765 6  
7663 7766 6  
7664 7767 6  
7665 7768 6  
7666 7769 6  
7667 7770 6  
7668 7771 6  
7669 7772 6  
7670 7773 6  
7671 7774 6  
7672 7775 6  
7673 7776 6  
7674 7777 6  
7675 7778 6  
7676 7779 6  
7677 7780 7  
7678 7781 7  
7679 7782 7  
7680 7783 6  
7681 7784 6  
7682 7785 5  
7683 7786 5  
7684 7787 4  
7685 7788 4  
7686 7789 4  
7687 7790 4  
7688 7791 4  
7689 7792 4  
7690 7793 4  
7691 7794 4  
7692 7795 4  
7693 7796 4  
7694 7797 4  
7695 7798 4  
7696 7799 4  
7697 7800 4  
7698 7801 4  
7699 7802 4  
7700 7803 4  
7701 7804 4  
7702 7805 4  
7703 7806 4  
7704 7807 4  
7705 7808 4

THEN  
BEGIN

! Check to see if the target is unsigned data type.

IF (.RIGHT\_ARG[DBG\$B\_VALUE\_DTYPE] EQL DSC\$K\_DTYPE\_WU OR  
.RIGHT\_ARG[DBG\$B\_VALUE\_DTYPE] EQL DSC\$K\_DTYPE\_LU OR  
.RIGHT\_ARG[DBG\$B\_VALUE\_DTYPE] EQL DSC\$K\_DTYPE\_QU OR  
.RIGHT\_ARG[DBG\$B\_VALUE\_DTYPE] EQL DSC\$K\_DTYPE\_NU) AND  
(.LEFT\_ARG[DBG\$B\_VALUE\_DTYPE] EQL DSC\$K\_DTYPE\_P)

THEN

BEGIN

LOCAL DIG, SIGN: BYTE, SIGN\_POS: REF VECTOR[,BYTE];

! Find out the sign position in number string.

DIG = .LEFT\_ARG[DBG\$W\_VALUE\_LENGTH];  
SIGN\_POS = .LEFT\_ARG[DBG\$L\_VALUE\_POINTER]  
+ (.DIG 7 2 + 1) - 1;

! Make it into positive.

SIGN = .SIGN\_POS[0] AND %X'0D';

IF .SIGN EQL %X'0D'

THEN

BEGIN

SIGN\_POS[0] = .SIGN\_POS[0] AND %X'FE';

SIGNAL(DBG\$\_IVALOUTBND, 1, OPERATOR[DBG\$B\_OPLEN]);

END;

END;

END;

! The following case is done for COBOL only. In cobol,  
if we do DEP sc2v2=999.888, sc2v2 is (SD, W with digits 4 and  
scaled -2). the answer we want is 99.88, instead of -310.xx.  
However if there is no digits specified in the Descriptor  
the base digits used is 5 then you'll get -310.xx as an  
answer.

The following code is a hack to force the type convertor  
to do the right thing for cobol, for example, if we deposit  
into a SD integer type, the way we make the type convertor to  
do the right thing is to create an intermediate data type of  
(SD, P) for left hand side as well, so the path looks like:  
right hand side converts (SD, P), creates a (SD, P) descriptor  
for left hand side, performs conversion from right (SD, P) to  
left (SD, P), final converts right (SD, P) to right hand side.

IF .DBG\$GB\_LANGUAGE EQL DBG\$K\_COBOL OR

.DBG\$GB\_LANGUAGE EQL DBG\$K\_RPG

THEN

```

7706      7809  5
7707      7810  5
7708      7811  5
7709      7812  6
7710      7813  6
7711      7814  6
7712      7815  6
7713      7816  6
7714      7817  6
7715      7818  6
7716      7819  6
7717      7820  6
7718      7821  6
7719      7822  6
7720      7823  6
7721      7824  6
7722      7825  6
7723      7826  6
7724      7827  6
7725      7828  6
7726      7829  6
7727      7830  6
7728      7831  7
7729      7832  7
7730      7833  6
7731      7834  5
7732      7835  4
7733      7836  4
7734      7837  4
7735      7838  4
7736      7839  4
7737      7840  4
7738      7841  4
7739      7842  4
7740      7843  4
7741      7844  4
7742      7845  4
7743      7846  4
7744      7847  4
7745      7848  3
7746      7849  3
7747      7850  3
7748      7851  3
7749      7852  4
7750      7853  4
7751      7854  4
7752      7855  5
7753      7856  5
7754      7857  5
7755      7858  5
7756      7859  5
7757      7860  5
7758      7861  5
7759      7862  5
7760      7863  5
7761      7864  4
7762      7865  4

      BEGIN
      IF .RIGHT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD
      THEN
          BEGIN

              ! Get the SD, P form intermediate data for the left hand side.
              ! Create a place holder. Note, the intermediate may not
              ! be created, for example, if we deposit P-->P, there
              ! is no need for it. So the NEW_RIGHT_ARG may be the
              ! same as RIGHT_ARG.
              NEW_RIGHT_ARG = INTMED_DATA_FOR_DEP(.RIGHT_ARG, DSC$K_DTYPE_P,
              TRUE);

              ! Perform convert from intermediate to intermediate. The last
              ! conversion is convert the intermediate to original left
              ! hand side data type.
              IF .NEW_RIGHT_ARG NEQ .RIGHT_ARG
              THEN
                  BEGIN
                      LEFT_ARG = DBG$TYPE_CONV(.LEFT_ARG, .NEW_RIGHT_ARG);
                      END;
                  END;
              END;

              RIGHT_ARG = DBG$TYPE_CONV (.LEFT_ARG, .RIGHT_ARG);

              IF .RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT AND
              .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DBG$K_DTYPE_PICT AND
              .DBG$B_LANGUAGE EQL DBG$K_COBOL
              THEN
                  RIGHT_ARG[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_T;

              RETURN .RIGHT_ARG;
              END

          ELSE
              SIGNAL(DBG$OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN])

          ELSE
              BEGIN
                  IF .DBG$B_LANGUAGE EQL DBG$K_BASIC
                  THEN
                      BEGIN
                          IF .NEW_LEFT_TYPE EQL DSC$K_DTYPE_P
                          THEN
                              MAP_PACKED (NEW_LEFT_TYPE, .LEFT_ARG[DBG$W_VALUE_LENGTH], .NEW_RIGHT_TYPE)
                          ELSE
                              IF .NEW_RIGHT_TYPE EQL DSC$K_DTYPE_P
                              THEN
                                  MAP_PACKED (NEW_RIGHT_TYPE, .RIGHT_ARG[DBG$W_VALUE_LENGTH], .NEW_LEFT_TYPE);
                              END;
                          END;
                      END;
                  END;
              END;
          END;
      END;
```

```
7763 7866 4      IF .MAX_DEPTH EQL FIND JOIN(
7764 7867 4      .NEW_LEFT_TYPE, .NEW_RIGHT_TYPE,
7765 7868 4      .NEW_LEFT_TYPE, .NEW_RIGHT_TYPE,
7766 7869 4      ROUT_TBL_INDEX,
7767 7870 4      0, .MAX_DEPTH,
7768 7871 4      .HIER_TBL, .HIER_TBL_SIZE,
7769 7872 4      .INCOMP_TBL, .INCOMP_TBL_SIZE,
7770 7873 4      .ROUT_TBL, .ROUT_TBL_SIZE)
7771 7874 4      THEN
7772 7875 4      SIGNAL(DBG$OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
7773 7876 4
7774 7877 4      END;
7775 7878 4
7776 7879 4      END
7777 7880 4
7778 7881 4      ELSE
7779 7882 4      BEGIN
7780 7883 4      IF .MAX_DEPTH EQL FIND_PATH(
7781 7884 4      .NEW_LEFT_TYPE, .NEW_LEFT_TYPE,
7782 7885 4      ROUT_TBL_INDEX,
7783 7886 4      .HIER_TBL, .HIER_TBL_SIZE,
7784 7887 4      .INCOMP_TBL, .INCOMP_TBL_SIZE,
7785 7888 4      .ROUT_TBL, .ROUT_TBL_SIZE)
7786 7889 4      THEN
7787 7890 4      SIGNAL(DBG$OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
7788 7891 4
7789 7892 4      END;
7790 7893 4
7791 7894 4
7792 7895 4      ! If one of the operand has T, VT, V, or VU in relational and arithmetic
7793 7896 4      ! operation in PLI. The target type for T, VT, V, or VU depends on
7794 7897 4      ! the other operand. This is not what we do normally. So, we use
7795 7898 4      ! the normal way to validate the operand, and then use this routine
7796 7899 4      ! to modify the other information. This is not the best way to do
7797 7900 4      ! it, but there is no other way I can think of is easier.
7798 7901 4
7799 7902 4      IF .BINARY_FLAG AND
7800 7903 4      .DBG$GB_LANGUAGE EQL DBG$K_PLI
7801 7904 4      THEN
7802 7905 4      MODIFY_PLI TARGET TYPE(.OPERATOR, .LEFT_TYPE, .RIGHT_TYPE,
7803 7906 4      .NEW_LEFT_TYPE, .NEW_RIGHT_TYPE, ROUT_TBL_INDEX,
7804 7907 4      .HIER_TBL, .HIER_TBL_SIZE,
7805 7908 4      .INCOMP_TBL, .INCOMP_TBL_SIZE,
7806 7909 4      .ROUT_TBL, .ROUT_TBL_SIZE);
7807 7910 4
7808 7911 4
7809 7912 4      ! Check for unconverted bit.
7810 7913 4
7811 7914 4      IF .LEFT_ARG[DBG$V_DHDR_UNCVT]
7812 7915 4      THEN
7813 7916 4      LEFT_ARG = DBG$CONV_TEXT_VALUE(.LEFT_ARG, .LEFT_ARG, .NEW_LEFT_TYPE);
7814 7917 4
7815 7918 4
7816 7919 4      ! For the left operand, check whether we need to convert to a new type.
7817 7920 4      ! If so, allocate space to hold the result value descriptor, and then
7818 7921 4      ! call a routine to do the conversion.
7819 7922 4
```

```

7820 7923 2
7821 7924 2
7822 7925 2
7823 7926 2
7824 7927 2
7825 7928 2
7826 7929 2
7827 7930 2
7828 7931 2
7829 7932 2
7830 7933 2
7831 7934 2
7832 7935 2
7833 7936 2
7834 7937 2
7835 7938 2
7836 7939 2
7837 7940 2
7838 7941 2
7839 7942 2
7840 7943 2
7841 7944 2
7842 7945 2
7843 7946 2
7844 7947 2
7845 7948 2
7846 7949 2
7847 7950 2
7848 7951 2
7849 7952 2
7850 7953 2
7851 7954 2
7852 7955 2
7853 7956 2
7854 7957 2
7855 7958 2
7856 7959 2
7857 7960 2
7858 7961 2
7859 7962 2
7860 7963 2
7861 7964 2
7862 7965 2
7863 7966 2
7864 7967 2
7865 7968 2
7866 7969 2
7867 7970 2
7868 7971 2
7869 7972 2
7870 7973 2
7871 7974 2
7872 7975 2
7873 7976 2
7874 7977 2
7875 7978 2
7876 7979 2

IF .LEFT_TYPE NEQ .NEW_LEFT_TYPE
THEN
  BEGIN
    LENGTH = GET_DATA_LENGTH(.LEFT_TYPE, .NEW_LEFT_TYPE,
      .LEFT_ARG[DBG$W_VALUE_LENGTH]);
    SD_CLASS_FLAG = FALSE;
    IF .DBG$GB_LANGUAGE EQL DBG$K_COBOL OR
      .DBG$GB_LANGUAGE EQL DBG$K_RPG
    THEN
      BEGIN
        IF .LEFT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD
        THEN
          SD_CLASS_FLAG = TRUE;
        END;
      NEW_LEFT_ARG = MAKE_VAL_DESC(.NEW_LEFT_TYPE,
        .LENGTH,
        .LEFT_ARG,
        .SD_CLASS_FLAG);

      IF .NEW_LEFT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD
      THEN
        BEGIN

          ! If left_arg is one of the floating-point data type, we need to
          ! get the exponent value and that is the scaling factor. For
          ! example F --> P, F = 0.1234567E+04 --> P = 1234.567.

          IF .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_F OR
            .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_D OR
            .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_G OR
            .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_H
          THEN
            BEGIN
              NEW_LEFT_ARG[DBG$B_VALUE_SCALE] = GET_SCALE(.LEFT_ARG, DIGITS);
              NEW_LEFT_ARG[DBG$B_VALUE_DIGITS] = .DIGITS;
              NEW_LEFT_ARG[DBG$W_VALUE_LENGTH] = .DIGITS;
            END
          ELSE
            NEW_LEFT_ARG[DBG$B_VALUE_SCALE] = .LEFT_ARG[DBG$B_VALUE_SCALE];
          END;
        NEW_LEFT_ARG = DBG$TYPE_CONV (.LEFT_ARG, .NEW_LEFT_ARG);
      END

      ! If there is no conversion to be done, just copy the left arg
      ! into NEW_LEFT_ARG.
    ELSE
      NEW_LEFT_ARG = .LEFT_ARG;

      ! Check for unconverted bit.

```

7877  
7878  
7879  
7880  
7881  
7882  
7883  
7884  
7885  
7886  
7887  
7888  
7889  
7890  
7891  
7892  
7893  
7894  
7895  
7896  
7897  
7898  
7899  
7900  
7901  
7902  
7903  
7904  
7905  
7906  
7907  
7908  
7909  
7910  
7911  
7912  
7913  
7914  
7915  
7916  
7917  
7918  
7919  
7920  
7921  
7922  
7923  
7924  
7925  
7926  
7927  
7928  
7929  
7930  
7931  
7932  
7933

7980  
7981  
7982  
7983  
7984  
7985  
7986  
7987  
7988  
7989  
7990  
7991  
7992  
7993  
7994  
7995  
7996  
7997  
7998  
7999  
8000  
8001  
8002  
8003  
8004  
8005  
8006  
8007  
8008  
8009  
8010  
8011  
8012  
8013  
8014  
8015  
8016  
8017  
8018  
8019  
8020  
8021  
8022  
8023  
8024  
8025  
8026  
8027  
8028  
8029  
8030  
8031  
8032  
8033  
8034  
8035  
8036

```
IF .BINARY_FLAG
THEN
  IF .RIGHT_ARG[DBG$V_DHDR_UNCVT]
  THEN
    RIGHT_ARG = DBG$CONV_TEXT_VALUE(.RIGHT_ARG, .RIGHT_ARG, .NEW_RIGHT_TYPE);

    ! If this is a binary operator, do the same for the right argument.
  !
  IF .BINARY_FLAG
  THEN
    IF .RIGHT_TYPE NEQ .NEW_RIGHT_TYPE
    THEN
      BEGIN
        LENGTH = GET_DATA_LENGTH(.RIGHT_TYPE, .NEW_RIGHT_TYPE,
          .RIGHT_ARG[DBG$W_VALUE_LENGTH]);
        SD_CLASS_FLAG = FALSE;
        IF .DBG$GB_LANGUAGE EQL DBG$K_COBOL OR
          .DBG$GB_LANGUAGE EQL DBG$K_RPG
        THEN
          BEGIN
            IF .RIGHT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD
            THEN
              SD_CLASS_FLAG = TRUE;
            END;
          NEW_RIGHT_ARG = MAKE_VAL_DESC(.NEW_RIGHT_TYPE,
            .LENGTH,
            .RIGHT_ARG,
            .SD_CLASS_FLAG);

          IF .NEW_RIGHT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD
          THEN
            BEGIN
              ! If right_arg is one of the floating-point data type, we need to
              ! get the exponent value and that is the scaling factor. For
              ! example F --> P, F = 0.1234567E+04 --> P = 1234.567.
              IF .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_F OR
                .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_D OR
                .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_G OR
                .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_H
              THEN
                BEGIN
                  NEW_RIGHT_ARG[DBG$B_VALUE_SCALE] = GET_SCALE(.RIGHT_ARG, DIGITS);
                  NEW_RIGHT_ARG[DBG$B_VALUE_DIGITS] = .DIGITS;
                  NEW_RIGHT_ARG[DBG$W_VALUE_LENGTH] = .DIGITS;
                END
              ELSE
                NEW_RIGHT_ARG[DBG$B_VALUE_SCALE] = .RIGHT_ARG[DBG$B_VALUE_SCALE];
              END;
            NEW_RIGHT_ARG = DBG$TYPE_CONV (.RIGHT_ARG, .NEW_RIGHT_ARG);
          END
```

```

      ELSE
        NEW_RIGHT_ARG = .RIGHT_ARG;

ROUT_INDEX = .ROUT_TBL[.ROUT_TBL_INDEX, ORTSW ROUT];
TYPEID_INDEX = .ROUT_TBL[.ROUT_TBL_INDEX, ORTSW TYPEID ROUT];
RESULT_TYPE = .ROUT_TBL[.ROUT_TBL_INDEX, ORTSW_RESULT_TYPE];
IF (.NEW_LEFT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD) OR
   (.BINARY_FLAG AND (.NEW_RIGHT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD))
THEN
  SD_CLASS_FLAG = TRUE
ELSE
  SD_CLASS_FLAG = FALSE;

LENGTH = 0;
IF .ROUT_INDEX EQL ORTSK_CONCAT_T_T OR
   .ROUT_INDEX EQL ORTSK_CONCAT_TF_TF
THEN
  LENGTH = .NEW_LEFT_ARG[DBG$W_VALUE_LENGTH] +
    .NEW_RIGHT_ARG[DBG$W_VALUE_LENGTH];

IF .LENGTH EQL 0
THEN
  BEGIN
    IF .RESULT_TYPE EQL DSC$K_DTYPE_V OR
       .RESULT_TYPE EQL DSC$K_DTYPE_VU OR
       .RESULT_TYPE EQL DSC$K_DTYPE_SV OR
       .RESULT_TYPE EQL DSC$K_DTYPE_SVU
    THEN
      BEGIN
        LENGTH = .NEW_LEFT_ARG[DBG$W_VALUE_LENGTH];
        IF .BINARY_FLAG
        THEN
          BEGIN
            IF .NEW_RIGHT_ARG[DBG$W_VALUE_LENGTH] GTR
               .NEW_LEFT_ARG[DBG$W_VALUE_LENGTH]
            THEN
              LENGTH = .NEW_RIGHT_ARG[DBG$W_VALUE_LENGTH];
            END;
          END
        ELSE
          LENGTH = DBG$NUM_BYTES(.RESULT_TYPE);
        END;
      END
    ELSE
      LENGTH = DBG$NUM_BYTES(.RESULT_TYPE);
    END;

  RESULT = MAKE_VAL_DESC(.RESULT_TYPE,
                        LENGTH,
                        0,
                        .SD_CLASS_FLAG);

  ! Make sure we did not turn SD_CLASS_FLAG on for the following
  ! data types. (in order words, ignore the incoming data types).
  CASE .RESULT[DBG$B_VALUE_DTYPE] FROM DSC$K_DTYPE_LOWEST
```

```
7991 8094
7992 8095
7993 8096
7994 8097
7995 8098
7996 8099
7997 8100
7998 8101
7999 8102
8000 8103
8001 8104
8002 8105
8003 8106
8004 8107
8005 8108
8006 8109
8007 8110
8008 8111
8009 8112
8010 8113
8011 8114
8012 8115
8013 8116
8014 8117
8015 8118
8016 8119
8017 8120
8018 8121
8019 8122
8020 8123
8021 8124
8022 8125
8023 8126
8024 8127
8025 8128
8026 8129
8027 8130
8028 8131
8029 8132
8030 8133
8031 8134
8032 8135
8033 8136
8034 8137
8035 8138
```

```
TO DSCSK_DTYPE_HIGHEST OF
SET
[DSCSK_DTYPE_F, DSCSK_DTYPE_D, DSCSK_DTYPE_G, DSCSK_DTYPE_H,
DSCSK_DTYPE_FC, DSCSK_DTYPE_DC, DSCSK_DTYPE_GC, DSCSK_DTYPE_HC]:
RESULT[DBG$B_VALUE_CLASS] = DSCSK_CLASS_5;
[INRANGE, OUTRANGE]:
0;
TES;

: Call the routine which performs the operation. We pass in
: the routine index, which is used as a case index inside the
: routine, the addresses of the operand value descriptors,
: and the address of a longword containing a pointer to the result
: value descriptor. The reason for the extra level of indirection
: for the result is that sometimes DBG$PERFORM_OPERATOR will construct
: a new descriptor instead of using the one we pass in.
DBG$PERFORM_OPERATOR (.OPERATOR,
:ROUT_INDEX,
:NEW_LEFT_ARG,
: (IF .BINARY_FLAG
: THEN .NEW_RIGHT_ARG
: ELSE 0),
:RESULT);

: Call TYPEID routine to do type check on non-atomic data types. And
: also fix up the Result Value Descriptor.
IF .TYPEID_INDEX NEQ 0
THEN
BEGIN
IF NOT DBG$PERFORM_TYPEID_CHECK(.TYPEID_INDEX, .NEW_LEFT_ARG,
:NEW_RIGHT_ARG, .RESULT)
THEN
SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
END;

: Return a pointer to the RESULT value descriptor.
RETURN .RESULT;
END;
```

```
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 31 05C8E P.ALS: .ASCII \1DBG$EVALOP\<92>\DBG$EVAL_LANG_OPERATOR,\
41 52 45 50 4F 5F 47 4E 41 4C 5F 4C 41 56 45 05C9D
20 79 63 6E 65 74 73 69 73 6E 6F 63 6E 69 20 05C80 .ASCII \ inconsistency 1\
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 31 05CBF
41 52 45 50 4F 5F 47 4E 41 4C 5F 4C 41 56 45 05CC0 P.ALT: .ASCII \1DBG$EVALOP\<92>\DBG$EVAL_LANG_OPERATOR,\
05CCF
```

```
.PSECT DBG$PLIT, NOWRT, SHR, PIC, 0
```

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        |        |   |  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|--------|--------|---|--|
| 20 | 79 | 63 | 6E | 65 | 74 | 73 | 69 | 73 | 6E | 6F | 2C | 52 | 4F | 54 | 05CDE |        |        |   |  |
| 24 | 47 | 43 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 31 | 05CE2 |        | .ASCII | \ inconsistency 2\                          |  |
| 41 | 52 | 45 | 50 | 4F | 5F | 47 | 4E | 41 | 4C | 5F | 4C | 41 | 56 | 45 | 05CF1 | P.ALU: | .ASCII | \1DBGEVALOP\<92>\DBG\$EVAL_LANG_OPERATOR,\  |  |
| 20 | 79 | 63 | 6E | 65 | 74 | 73 | 69 | 73 | 6E | 6F | 2C | 52 | 4F | 54 | 05D01 |        |        |   |  |
| 24 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 33 | 05D10 |        | .ASCII | \ inconsistency 3\                          |  |
| 41 | 52 | 45 | 50 | 4F | 5F | 47 | 4E | 41 | 4C | 5F | 4C | 41 | 56 | 45 | 05D23 | P.ALV: | .ASCII | \6DBGEVALOP\<92>\DBG\$EVAL_LANG_OPERATOR \  |  |
| 6D | 20 | 65 | 6C | 62 | 61 | 74 | 20 | 65 | 6E | 69 | 74 | 75 | 6F | 72 | 05D42 |        | .ASCII | \routine table missing\                     |  |
| 45 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 34 | 05D55 |        |        |   |  |
| 54 | 41 | 52 | 45 | 50 | 4F | 5F | 47 | 4E | 41 | 4C | 5F | 4C | 41 | 56 | 05D5B | P.ALW: | .ASCII | \4DBGEVALOP\<92>\DBG\$EVAL_LANG_OPERATOR, \ |  |
| 61 | 66 | 20 | 72 | 64 | 64 | 61 | 20 | 6F | 74 | 20 | 6D | 69 | 72 | 70 | 05D6A |        |        |   |  |
| 24 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 34 | 05D79 |        | .ASCII | \prim to addr failed\                       |  |
| 41 | 52 | 45 | 50 | 4F | 5F | 47 | 4E | 41 | 4C | 5F | 4C | 41 | 56 | 45 | 05D7D | P.ALX: | .ASCII | \4DBGEVALOP\<92>\DBG\$EVAL_LANG_OPERATOR,\  |  |
| 61 | 66 | 20 | 6C | 61 | 76 | 20 | 6F | 74 | 20 | 6D | 69 | 72 | 70 | 20 | 05D8C |        | .ASCII | \ prim to val failed\                       |  |
| 45 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 34 | 05D90 |        |        |   |  |
| 54 | 41 | 52 | 45 | 50 | 4F | 5F | 47 | 4E | 41 | 4C | 5F | 4C | 41 | 56 | 05D9F | P.ALY: | .ASCII | \4DBGEVALOP\<92>\DBG\$EVAL_LANG_OPERATOR, \ |  |
| 61 | 66 | 20 | 72 | 64 | 64 | 61 | 20 | 6F | 74 | 20 | 6D | 69 | 72 | 70 | 05DAE |        | .ASCII | \prim to addr failed\                       |  |
| 45 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 34 | 05DB2 |        |        |   |  |
| 54 | 41 | 52 | 45 | 50 | 4F | 5F | 47 | 4E | 41 | 4C | 5F | 4C | 41 | 56 | 05DC1 | P.ALZ: | .ASCII | \4DBGEVALOP\<92>\DBG\$EVAL_LANG_OPERATOR, \ |  |
| 61 | 66 | 20 | 72 | 64 | 64 | 61 | 20 | 6F | 74 | 20 | 6D | 69 | 72 | 70 | 05DC5 |        | .ASCII | \prim to addr failed\                       |  |
| 45 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 34 | 05DD4 |        |        |   |  |
| 54 | 41 | 52 | 45 | 50 | 4F | 5F | 47 | 4E | 41 | 4C | 5F | 4C | 41 | 56 | 05DE3 | P.AMA: | .ASCII | \4DBGEVALOP\<92>\DBG\$EVAL_LANG_OPERATOR,\  |  |
| 61 | 66 | 20 | 72 | 64 | 64 | 61 | 20 | 6F | 74 | 20 | 6D | 69 | 72 | 70 | 05DE7 |        | .ASCII | \prim to val failed\                        |  |
| 45 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 34 | 05DF6 |        |        |   |  |
| 54 | 41 | 52 | 45 | 50 | 4F | 5F | 47 | 4E | 41 | 4C | 5F | 4C | 41 | 56 | 05DFA | P.AMB: | .ASCII | \4DBGEVALOP\<92>\DBG\$EVAL_LANG_OPERATOR, \ |  |
| 61 | 66 | 20 | 72 | 64 | 64 | 61 | 20 | 6F | 74 | 20 | 6D | 69 | 72 | 70 | 05E09 |        | .ASCII | \prim to addr failed\                       |  |
| 45 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 34 | 05E18 |        |        |   |  |
| 54 | 41 | 52 | 45 | 50 | 4F | 5F | 47 | 4E | 41 | 4C | 5F | 4C | 41 | 56 | 05E2B |        |        |   |  |
| 61 | 66 | 20 | 72 | 64 | 64 | 61 | 20 | 6F | 74 | 20 | 6D | 69 | 72 | 70 | 05E2F |        | .ASCII | \ prim to val failed\                       |  |
| 45 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 34 | 05E3E |        |        |   |  |
| 54 | 41 | 52 | 45 | 50 | 4F | 5F | 47 | 4E | 41 | 4C | 5F | 4C | 41 | 56 | 05E4D |        | .ASCII | \4DBGEVALOP\<92>\DBG\$EVAL_LANG_OPERATOR, \ |  |
| 61 | 66 | 20 | 72 | 64 | 64 | 61 | 20 | 6F | 74 | 20 | 6D | 69 | 72 | 70 | 05E51 |        |        |   |  |
| 45 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 34 | 05E60 |        |        |   |  |
| 54 | 41 | 52 | 45 | 50 | 4F | 5F | 47 | 4E | 41 | 4C | 5F | 4C | 41 | 56 | 05E64 |        |        |   |  |
| 61 | 66 | 20 | 72 | 64 | 64 | 61 | 20 | 6F | 74 | 20 | 6D | 69 | 72 | 70 | 05E73 |        | .ASCII | \prim to addr failed\                       |  |
| 45 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 34 | 05E82 |        |        |   |  |
| 54 | 41 | 52 | 45 | 50 | 4F | 5F | 47 | 4E | 41 | 4C | 5F | 4C | 41 | 56 | 05E86 |        |        |   |  |
| 61 | 66 | 20 | 72 | 64 | 64 | 61 | 20 | 6F | 74 | 20 | 6D | 69 | 72 | 70 | 05E95 |        |        |   |  |

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

OFFC 00000

.ENTRY DBG\$EVAL\_LANG\_OPERATOR, Save R2,R3,R4,R5,-  
R6,R7,R8,R9,R10,R11

7068

SUBL2 #36, \$P

CLRL R3

7164

CMPZV #16, #16, @OPERATOR, #4

BNEQ 1\$

INCL R3

BRB 3\$

CMPZV #16, #16, @OPERATOR, #5

7165

04 04 BC

5E

24 C2 00002

53 D4 00005

10 ED 00007

04 12 0000D

53 D6 0000F

0B 11 00011

10 ED 00013

05 04 BC

10

1\$:

|  |  |  |  |  |           |    |       |       |        |                                  |                          |      |
|--|--|--|--|--|-----------|----|-------|-------|--------|----------------------------------|--------------------------|------|
|  |  |  |  |  | 03        | 13 | 00019 |       | BEQL   | 3\$                              |                          |      |
|  |  |  |  |  | 008A      | 31 | 0001B | 2\$:  | BRW    | 12\$                             |                          |      |
|  |  |  |  |  | AC        | D0 | 0001E | 3\$:  | MOVL   | LEFT_ARG, R2                     |                          | 7168 |
|  |  |  |  |  | 05        | E1 | 00022 |       | BBC    | #5, 2(R2), 2\$                   |                          |      |
|  |  |  |  |  | 0C        | D5 | 00027 |       | TSTL   | RIGHT_ARG                        |                          | 7175 |
|  |  |  |  |  | 15        | 13 | 0002A |       | BEQL   | 4\$                              |                          |      |
|  |  |  |  |  | 00000000' | EF | 9F    | 0002C | PUSHAB | P.AL5                            |                          | 7177 |
|  |  |  |  |  | 00028362  | 01 | DD    | 00032 | PUSHL  | #1                               |                          |      |
|  |  |  |  |  |           | 8F | DD    | 00034 | PUSHL  | #164706                          |                          |      |
|  |  |  |  |  | 00000000G | 03 | FB    | 0003A | CALLS  | #3, LIB\$SIGNAL                  |                          |      |
|  |  |  |  |  | 79        | 8F |       |       | CMPB   | 2(R2), #121                      |                          | 7183 |
|  |  |  |  |  |           | 02 | A2    | 91    | 00041  | 4\$:                             |                          |      |
|  |  |  |  |  |           | 07 | 13    | 00046 | BEQL   | 5\$                              |                          |      |
|  |  |  |  |  |           | 02 | A2    | 91    | 00048  | CMPB                             | 2(R2), #131              | 7184 |
|  |  |  |  |  |           | 15 | 12    | 0004D | BNEQ   | 6\$                              |                          |      |
|  |  |  |  |  | 00000000' | EF | 9F    | 0004F | 5\$:   | PUSHAB                           | P.AL7                    | 7186 |
|  |  |  |  |  |           | 01 | DD    | 00055 | PUSHL  | #1                               |                          |      |
|  |  |  |  |  | 00028362  | 8F | DD    | 00057 | PUSHL  | #164706                          |                          |      |
|  |  |  |  |  | 00000000G | 03 | FB    | 0005D | CALLS  | #3, LIB\$SIGNAL                  |                          |      |
|  |  |  |  |  | 00        | 10 | ED    | 00064 | 6\$:   | CMPZV                            | #16, #16, @OPERATOR, #5  | 7193 |
|  |  |  |  |  |           | 0A | 12    | 0006A | BNEQ   | 7\$                              |                          |      |
|  |  |  |  |  | 0042      | 8F |       |       | CMPW   | 18(R2), #66                      |                          | 7196 |
|  |  |  |  |  |           | 0D | 12    | 00072 | BNEQ   | 8\$                              |                          |      |
|  |  |  |  |  |           | 12 | 11    | 00074 | BRB    | 9\$                              |                          | 7198 |
|  |  |  |  |  |           | 53 | E9    | 00076 | 7\$:   | BLBC                             | R3, 10\$                 | 7204 |
|  |  |  |  |  | 0042      | 8F |       |       | CMPW   | 18(R2), #66                      |                          | 7207 |
|  |  |  |  |  |           | A2 | B1    | 00079 | BNEQ   | 9\$                              |                          |      |
|  |  |  |  |  |           | 07 | 12    | 0007F | MOVZBW | #66, 18(R2)                      |                          | 7209 |
|  |  |  |  |  |           | 8F | 9B    | 00081 | BRB    | 11\$                             |                          |      |
|  |  |  |  |  |           | 1C | 11    | 00086 | MOVZBW | #67, 18(R2)                      |                          | 7211 |
|  |  |  |  |  |           | 8F | 9B    | 00088 | BRB    | 11\$                             |                          | 7204 |
|  |  |  |  |  |           | 15 | 11    | 0008D | PUSHAB | P.ALU                            |                          | 7215 |
|  |  |  |  |  | 00000000' | EF | 9F    | 0008F | 10\$:  | PUSHL                            | #1                       |      |
|  |  |  |  |  |           | 01 | DD    | 00095 | PUSHL  | #164706                          |                          |      |
|  |  |  |  |  | 00028362  | 8F | DD    | 00097 | CALLS  | #3, LIB\$SIGNAL                  |                          |      |
|  |  |  |  |  | 00000000G | 03 | FB    | 0009D | MOVL   | R2, R0                           |                          | 7217 |
|  |  |  |  |  |           | 50 | D0    | 000A4 | 11\$:  | RET                              |                          |      |
|  |  |  |  |  |           | 04 | 000A7 |       | CMPZV  | #16, #16, @OPERATOR, #66         |                          | 7227 |
|  |  |  |  |  |           | 10 | ED    | 000AB | 12\$:  | BNEQ                             | 13\$                     |      |
|  |  |  |  |  |           | 08 | 12    | 000B2 | MOVAB  | DBG\$GL NEG SIGN_TOKEN, OPERATOR |                          | 7229 |
|  |  |  |  |  |           | 00 | 9E    | 000B4 | 13\$:  | CMPZV                            | #16, #16, @OPERATOR, #67 | 7231 |
|  |  |  |  |  |           | 10 | ED    | 000BC | BNEQ   | 14\$                             |                          |      |
|  |  |  |  |  |           | 08 | 12    | 000C6 | MOVAB  | DBG\$GL POS SIGN_TOKEN, OPERATOR |                          | 7233 |
|  |  |  |  |  |           | 00 | 9E    | 000C8 | 14\$:  | MOVL                             | #1, MAX_DEPTH            | 7238 |
|  |  |  |  |  | 00000000' | 01 | D0    | 000D0 | BLBC   | BLISS_BITSELECTION_FLAG1, 15\$   |                          | 7239 |
|  |  |  |  |  |           | 09 | E9    | 000D7 | MOVL   | #1, BLISS_BITSELECTION_FLAG2     |                          | 7241 |
|  |  |  |  |  | 00000000' | 01 | D0    | 000DE | BRB    | 16\$                             |                          |      |
|  |  |  |  |  |           | 06 | 11    | 000E5 | CLRL   | BLISS_BITSELECTION_FLAG2         |                          | 7243 |
|  |  |  |  |  | 00000000' | EF | D4    | 000E7 | 15\$:  | CLRL                             | BLISS_BITSELECTION_FLAG1 | 7244 |
|  |  |  |  |  | 00000000' | EF | D4    | 000ED | 16\$:  | CLRL                             | BLISS_INDIRECTION_FLAG   | 7245 |
|  |  |  |  |  | 00000000' | EF | D4    | 000F3 | EXTZV  | #16, #16, @OPERATOR, OPCODE      |                          | 7246 |
|  |  |  |  |  |           | 10 | EF    | 000F9 | ADDL3  | #12, OPERATOR, R11               |                          | 7247 |
|  |  |  |  |  |           | 0C | C1    | 000FF | MOVL   | R11, DBG\$GL_OPCODE_NAME         |                          |      |
|  |  |  |  |  |           | 5B | D0    | 00104 | CLRL   | R0                               |                          | 7248 |
|  |  |  |  |  |           | 50 | D4    | 0010B | CMPB   | @OPERATOR, #3                    |                          |      |
|  |  |  |  |  |           | 03 | 04    | BC    | 91     | 0010D                            |                          |      |
|  |  |  |  |  |           | 02 | 12    | 00111 | BNEQ   | 17\$                             |                          |      |
|  |  |  |  |  |           | 50 | D6    | 00113 | INCL   | R0                               |                          |      |
|  |  |  |  |  |           | 50 | 90    | 00115 | 17\$:  | MOVB                             | R0, BINARY_FLAG          |      |
|  |  |  |  |  | 0C        | AE |       |       |        |                                  |                          |      |

|    |           |           |      |           |        |                                      |      |
|----|-----------|-----------|------|-----------|--------|--------------------------------------|------|
| 58 | 52        | 04        | 78   | 00119     | ASHL   | #4, OPCODE, R8                       | 7249 |
| 51 | 58        | EF        | C1   | 0011D     | ADDL3  | OPINFO TABLE, R8, R1                 |      |
|    | 50        | EF        | 9E   | 00125     | MOVAB  | TABLEBASE, R0                        |      |
| 57 | 50        | 61        | C1   | 0012C     | ADDL3  | (R1), R0, ROUT_TBL                   |      |
|    | 50        | EF        | 9E   | 00130     | MOVAB  | TABLEBASE, R0                        | 7250 |
|    | 50        | 57        | D1   | 00137     | CMPL   | ROUT_TBL, R0                         |      |
|    |           | 26        | 12   | 0013A     | BNEQ   | 18\$,                                |      |
|    | 32        | 52        | D1   | 0013C     | CMPL   | OPCODE, #50                          | 7253 |
|    |           | 27        | 13   | 0013F     | BEQL   | 19\$,                                |      |
|    | 33        | 52        | D1   | 00141     | CMPL   | OPCODE, #51                          |      |
|    |           | 22        | 13   | 00144     | BEQL   | 19\$,                                |      |
|    | 38        | 52        | D1   | 00146     | CMPL   | OPCODE, #56                          | 7254 |
|    |           | 1D        | 13   | 00149     | BEQL   | 19\$,                                |      |
|    |           | 00000000' | EF   | 9F        | PUSHAB | P.ALV                                | 7256 |
|    |           | 00028362  | 01   | DD        | PUSHL  | #1                                   |      |
|    |           |           | 8F   | DD        | PUSHL  | #164706                              |      |
|    | 00000000G | 00        | 03   | FB        | CALLS  | #3, LIB\$SIGNAL                      |      |
|    |           |           | 06   | 11        | BRB    | 19\$,                                | 7250 |
| 04 | AE        | FC        | A7   | 02        | DIVL3  | #2, -4(ROUT_TBL), ROUT_TBL_SIZE      | 7259 |
|    |           |           | 50   | EF        | MOVAB  | OPINFO TABLE, R0                     | 7260 |
|    |           |           | 51   | EF        | MOVAB  | TABLEBASE, R1                        |      |
|    |           |           | 04   | A840      | PUSHAB | 4(R8)[R0]                            |      |
|    | 5A        |           | 51   | 9E        | ADDL3  | @(SP)+, R1, HIER_TBL                 |      |
|    |           |           | 51   | EF        | MOVAB  | TABLEBASE, R1                        | 7261 |
|    |           |           | 51   | 5A        | CMPL   | HIER_TBL, R1                         |      |
|    |           |           |      | 11        | BEQL   | 20\$,                                |      |
|    | 55        | FC        | AA   | 01        | ASHL   | #1, -4(HIER_TBL), HIER_TBL_SIZE      | 7264 |
|    |           | 00000000' | EF   | FF45      | MOVAB  | @MAX_DEPTH[HIER_TBL_SIZE], MAX_DEPTH | 7265 |
|    |           |           | 51   | EF        | MOVAB  | TABLEBASE, R1                        | 7267 |
|    |           |           |      | 08        | PUSHAB | 8(R8)[R0]                            |      |
|    | 59        |           | 51   | 9E        | ADDL3  | @(SP)+, R1, INCOMP_TBL               |      |
|    |           |           | 51   | EF        | MOVAB  | TABLEBASE, R1                        | 7268 |
|    |           |           | 51   | 59        | CMPL   | INCOMP_TBL, R1                       |      |
|    |           |           |      | 06        | BEQL   | 21\$,                                |      |
| 08 | AE        | FC        | A9   | 01        | ASHL   | #1, -4(INCOMP_TBL), INCOMP_TBL_SIZE  |      |
|    |           |           | 53   | 08        | MOVAB  | LEFT_ARG, R3                         | 7285 |
|    |           | 79        | 8F   | 02        | CMPL   | 2(R3), #121                          |      |
|    |           |           |      | 07        | BEQL   | 22\$,                                |      |
|    |           | 83        | 8F   | 02        | CMPL   | 2(R3), #131                          | 7286 |
|    |           |           |      | 55        | BNEQ   | 25\$,                                |      |
|    | 03        | 0C        | A840 | 00        | BBS    | #0, 12(R8)[R0], 23\$,                | 7303 |
|    |           |           |      | 00C2      | BRW    | 33\$,                                |      |
|    |           | 00000000' | EF   | 01        | MOVAB  | #1, BLISS INDIRECTION_FLAG           | 7306 |
|    |           |           | 07   | 00000000G | CMPL   | DBG\$GB_LANGUAGE, #7                 | 7312 |
|    |           |           |      | 3F        | BNEQ   | 26\$,                                |      |
|    |           |           | 01   | 06        | CMPL   | 6(R3), #1                            | 7313 |
|    |           |           |      | 39        | BNEQ   | 26\$,                                |      |
|    |           |           | 56   | 08        | MOVAB  | 8(R3), TYPEID                        | 7319 |
|    |           |           |      | 08        | PUSHAB | LEFT_ARG                             | 7323 |
|    |           |           |      | 08        | PUSHL  | #8                                   |      |
|    |           |           |      | 53        | PUSHL  | R3                                   |      |
|    | 00000000G | 00        | 03   | FB        | CALLS  | #3, DBG\$PRIM_TO_ADDR                |      |
|    |           |           | 15   | 50        | BLBS   | R0, 24\$,                            |      |
|    |           |           |      | 00000000' | PUSHAB | P.ALW                                | 7325 |
|    |           |           |      | 01        | PUSHL  | #1                                   |      |
|    |           |           |      | 00028362  | PUSHL  | #164706                              |      |
|    |           |           |      | 8F        | PUSHL  | #3, LIB\$SIGNAL                      |      |
|    | 00000000G | 00        | 03   | FB        | CALLS  |                                      |      |

|           |    |           |      |    |       |      |        |                            |      |
|-----------|----|-----------|------|----|-------|------|--------|----------------------------|------|
|           | 50 | 08        | AC   | D0 | 00217 | 24%: | MOVL   | LEFT_ARG, R0               | 7329 |
| 06        | A0 |           | 06   | 90 | 0021B |      | MOVB   | #6, 8(R0)                  |      |
| 08        | A0 |           | 56   | D0 | 0021F |      | MOVL   | TYPEID, 8(R0)              | 7330 |
|           |    |           | 0099 | 31 | 00223 | 25%: | BRW    | 35\$                       | 7312 |
|           | 09 | 04        | A3   | E9 | 00226 | 26%: | BLBC   | 4(R3), 27\$                | 7340 |
|           |    |           | 53   | DD | 0022A |      | PUSHL  | R3                         | 7342 |
| 00000000G | 00 |           | 01   | FB | 0022C |      | CALLS  | #1, DBG\$COLLECT           |      |
|           |    |           | 54   | D4 | 00233 | 27%: | CLRL   | SYMID                      | 7349 |
|           | 45 | 04        | A3   | E9 | 00235 |      | BLBC   | 4(R3), 32\$                | 7350 |
|           | 07 | 06        | A3   | 91 | 00239 |      | CMPB   | 6(R3), #7                  | 7353 |
|           |    |           | 0D   | 13 | 0023D |      | BEQL   | 28\$                       |      |
|           |    | 000287F8  | 8F   | DD | 0023F |      | PUSHL  | #165880                    | 7355 |
| 00000000G | 00 |           | 01   | FB | 00245 |      | CALLS  | #1, LIB\$SIGNAL            |      |
|           | 54 | 0C        | A3   | D0 | 0024C | 28%: | MOVL   | 12(R3), SYMID              | 7360 |
|           |    |           | 0D   | 12 | 00250 | 29%: | BNEQ   | 30\$                       | 7361 |
|           |    | 000287F8  | 8F   | DD | 00252 |      | PUSHL  | #165880                    |      |
| 00000000G | 00 |           | 01   | FB | 00258 |      | CALLS  | #1, LIB\$SIGNAL            |      |
|           | 01 | 14        | A4   | 91 | 0025F | 30%: | CMPB   | 20(SYMID), #1              | 7362 |
|           |    |           | 06   | 13 | 00263 |      | BEQL   | 31\$                       |      |
|           | 54 | 10        | A4   | D0 | 00265 |      | MOVL   | 16(SYMID), SYMID           | 7364 |
|           |    |           | E5   | 11 | 00269 |      | BRB    | 29\$                       | 7365 |
|           | 03 | 29        | A4   | 91 | 0026B | 31%: | CMPB   | 41(SYMID), #3              | 7367 |
|           |    |           | 0D   | 13 | 0026F |      | BEQL   | 32\$                       |      |
|           |    | 000287F8  | 8F   | DD | 00271 |      | PUSHL  | #165880                    | 7369 |
| 00000000G | 00 |           | 01   | FB | 00277 |      | CALLS  | #1, LIB\$SIGNAL            |      |
|           |    | 08        | AC   | 9F | 0027E | 32%: | PUSHAB | LEFT_ARG                   | 7372 |
|           | 7E | 7A        | 8F   | 9A | 00281 |      | MOVZBL | #122, -(SP)                |      |
|           |    |           | 53   | DD | 00285 |      | PUSHL  | R3                         |      |
| 00000000G | 00 |           | 03   | FB | 00287 |      | CALLS  | #3, DBG\$PRIM_TO_VAL       |      |
|           | 2E |           | 50   | E8 | 0028E |      | BLBS   | R0, 35\$                   |      |
|           |    | 00000000' | EF   | 9F | 00291 |      | PUSHAB | P.ALX                      | 7374 |
|           |    |           | 17   | 11 | 00297 |      | BRB    | 34\$                       |      |
|           |    | 08        | AC   | 9F | 00299 | 33%: | PUSHAB | LEFT_ARG                   | 7380 |
|           |    |           | 08   | DD | 0029C |      | PUSHL  | #8                         |      |
|           |    |           | 53   | DD | 0029E |      | PUSHL  | R3                         |      |
| 00000000G | 00 |           | 03   | FB | 002A0 |      | CALLS  | #3, DBG\$PRIM_TO_ADDR      |      |
|           | 15 |           | 50   | E8 | 002A7 |      | BLBS   | R0, 35\$                   |      |
|           |    | 00000000' | EF   | 9F | 002AA |      | PUSHAB | P.ALX                      | 7382 |
|           |    |           | 01   | DD | 002B0 | 34%: | PUSHL  | #1                         |      |
|           |    | 00028362  | 8F   | DD | 002B2 |      | PUSHL  | #164706                    |      |
| 00000000G | 00 |           | 03   | FB | 002B8 |      | CALLS  | #3, LIB\$SIGNAL            |      |
|           | 77 | 0C        | AE   | E9 | 002BF | 35%: | BLBC   | BINARY_FLAG, 39\$          | 7387 |
|           | 53 | 0C        | AC   | D0 | 002C3 |      | MOVL   | RIGHT_ARG, R3              | 7389 |
|           | 79 | 02        | A3   | 91 | 002C7 |      | CMPB   | 2(R3), #121                |      |
|           |    |           | 11   | 13 | 002CC |      | BEQL   | 36\$                       |      |
|           | 83 | 02        | A3   | 91 | 002CE |      | CMPB   | 2(R3), #131                | 7390 |
|           |    |           | 65   | 12 | 002D3 |      | BNEQ   | 39\$                       |      |
|           | 32 |           | 52   | D1 | 002D5 |      | CMPL   | OPCODE, #50                | 7391 |
|           |    |           | 60   | 13 | 002D8 |      | BEQL   | 39\$                       |      |
|           | 33 |           | 52   | D1 | 002DA |      | CMPL   | OPCODE, #51                |      |
|           |    |           | 5B   | 13 | 002DD |      | BEQL   | 39\$                       |      |
| 50        | 58 | 00000000' | EF   | C1 | 002DF | 36%: | ADDL3  | OPINFO_TABLE, R8, R0       | 7393 |
|           | 03 | 0C        | A0   | E8 | 002E7 |      | BLBS   | 12(R0), 37\$               |      |
|           |    |           | 00C2 | 31 | 002EB |      | BRW    | 47\$                       |      |
|           |    |           | 01   | D0 | 002EE | 37%: | MOVL   | #1, BLISS_INDIRECTION_FLAG | 7396 |
| 00000000' | EF |           | 00   | 91 | 002F5 |      | CMPB   | DBG\$GB_LANGUAGE, #7       | 7402 |
|           | 07 | 00000000G | 3F   | 12 | 002FC |      | BNEQ   | 40\$                       |      |

|           |           |      |    |       |        |                       |      |
|-----------|-----------|------|----|-------|--------|-----------------------|------|
| 01        | 06        | A3   | 91 | 002FE | CMPB   | 6(R3), #1             | 7403 |
|           |           | 39   | 12 | 00302 | BNEQ   | 40\$                  |      |
| 56        | 08        | A3   | 00 | 00304 | MOVL   | 8(R3), TYPEID         | 7409 |
|           | 0C        | AC   | 9F | 00308 | PUSHAB | RIGHT_ARG             | 7413 |
|           |           | 08   | DD | 0030B | PUSHL  | #8                    |      |
|           |           | 53   | DD | 0030D | PUSHL  | R3                    |      |
| 00000000G | 00        | 03   | FB | 0030F | CALLS  | #3, DBG\$PRIM_TO_ADDR |      |
|           | 15        | 50   | E8 | 00316 | BLBS   | R0, 38\$              |      |
|           | 00000000' | EF   | 9F | 00319 | PUSHAB | P.ALZ                 | 7415 |
|           |           | 01   | DD | 0031F | PUSHL  | #1                    |      |
|           | 00028362  | 8F   | DD | 00321 | PUSHL  | #164706               |      |
| 00000000G | 00        | 03   | FB | 00327 | CALLS  | #3, LIB\$SIGNAL       |      |
|           | 50        | AC   | DD | 0032E | MOVL   | RIGHT_ARG, R0         | 7419 |
| 06        | A0        | 06   | 90 | 00332 | MOVB   | #6, 6(R0)             |      |
| 08        | A0        | 56   | DD | 00336 | MOVL   | TYPEID, 8(R0)         | 7420 |
|           |           | 0099 | 31 | 0033A | BRW    | 49\$                  | 7402 |
|           | 09        | A3   | E9 | 0033D | BLBC   | 4(R3), 41\$           | 7431 |
|           | 04        | 53   | DD | 00341 | PUSHL  | R3                    | 7433 |
| 00000000G | 00        | 01   | FB | 00343 | CALLS  | #1, DBG\$COLLECT      |      |
|           |           | 54   | D4 | 0034A | CLRL   | SYMID                 | 7440 |
|           | 45        | A3   | E9 | 0034C | BLBC   | 4(R3), 46\$           | 7441 |
|           | 07        | A3   | 91 | 00350 | CMPB   | 6(R3), #7             | 7444 |
|           |           | 0D   | 13 | 00354 | BEQL   | 42\$                  |      |
|           | 000287F8  | 8F   | DD | 00356 | PUSHL  | #165880               | 7446 |
| 00000000G | 00        | 01   | FB | 0035C | CALLS  | #1, LIB\$SIGNAL       |      |
|           | 54        | A3   | DD | 00363 | MOVL   | 12(R3), SYMID         | 7451 |
|           |           | 0D   | 12 | 00367 | BNEQ   | 44\$                  | 7452 |
|           | 000287F8  | 8F   | DD | 00369 | PUSHL  | #165880               |      |
| 00000000G | 00        | 01   | FB | 0036F | CALLS  | #1, LIB\$SIGNAL       |      |
|           | 01        | A4   | 91 | 00376 | CMPB   | 20(SYMID), #1         | 7453 |
|           |           | 06   | 13 | 0037A | BEQL   | 45\$                  |      |
|           | 54        | A4   | DD | 0037C | MOVL   | 16(SYMID), SYMID      | 7455 |
|           |           | E5   | 11 | 00380 | BRB    | 43\$                  | 7456 |
|           | 03        | A4   | 91 | 00382 | CMPB   | 41(SYMID), #3         | 7458 |
|           |           | 0D   | 13 | 00386 | BEQL   | 46\$                  |      |
|           | 000287F8  | 8F   | DD | 00388 | PUSHL  | #165880               | 7460 |
| 00000000G | 00        | 01   | FB | 0038E | CALLS  | #1, LIB\$SIGNAL       |      |
|           |           | AC   | 9F | 00395 | PUSHAB | RIGHT_ARG             | 7463 |
|           | 7E        | 8F   | 9A | 00398 | MOVZBL | #122, -(SP)           |      |
|           |           | 53   | DD | 0039C | PUSHL  | R3                    |      |
| 00000000G | 00        | 03   | FB | 0039E | CALLS  | #3, DBG\$PRIM_TO_VAL  |      |
|           | 2E        | 50   | E8 | 003A5 | BLBS   | R0, 49\$              |      |
|           | 00000000' | EF   | 9F | 003A8 | PUSHAB | P.AMA                 | 7465 |
|           |           | 17   | 11 | 003AE | BRB    | 48\$                  |      |
|           |           | AC   | 9F | 003B0 | PUSHAB | RIGHT_ARG             | 7470 |
|           |           | 08   | DD | 003B3 | PUSHL  | #8                    |      |
|           |           | 53   | DD | 003B5 | PUSHL  | R3                    |      |
| 00000000G | 00        | 03   | FB | 003B7 | CALLS  | #3, DBG\$PRIM_TO_ADDR |      |
|           | 15        | 50   | E8 | 003BE | BLBS   | R0, 49\$              |      |
|           | 00000000' | EF   | 9F | 003C1 | PUSHAB | P.AMB                 | 7472 |
|           |           | 01   | DD | 003C7 | PUSHL  | #1                    |      |
|           | 00028362  | 8F   | DD | 003C9 | PUSHL  | #164706               |      |
| 00000000G | 00        | 03   | FB | 003CF | CALLS  | #3, LIB\$SIGNAL       |      |
|           |           | 54   | D4 | 003D6 | CLRL   | R4                    | 7477 |
|           | 32        | 52   | D1 | 003D8 | CPL    | OPCODE, #50           |      |
|           |           | 25   | 12 | 003DB | BNEQ   | 50\$                  |      |
|           |           | 54   | D6 | 003DD | INCL   | R4                    |      |

|       |    |                      |                |                            |                            |   |                                     |  |              |
|-------|----|----------------------|----------------|----------------------------|----------------------------|---|-------------------------------------|--|--------------|
| 1A    | 04 | 51<br>A1<br>50<br>14 | 08<br>0C<br>06 | AC<br>05<br>AC<br>A0<br>10 | DO<br>E1<br>DO<br>91<br>12 | 003DF<br>003E3<br>003E8<br>003EC<br>003F0 | MOVL<br>BBC<br>MOVL<br>CMPB<br>BNEQ | LEFT_ARG, R1<br>#5, Z(R1), 50\$<br>RIGHT_ARG, R0<br>6(R0), #20<br>50\$ | 7480<br>7483 |
|       | 08 | A1                   | 08             | A0<br>51                   | DO<br>DD                   | 003F2<br>003F7                            | MOVL<br>PUSHL                       | 8(R0), 8(R1)<br>R1   | 7486<br>7487 |
| F8B3  | CF |                      |                | 01                         | FB                         | 003F9                                     | CALLS                               | #1, DBG\$CONV_TRFA_VALUE   |              |
| 08    | AC |                      |                | 50                         | DO                         | 003FE                                     | MOVL                                | R0, LEFT_ARG   |              |
|       | 08 | 00000000G            |                | 00                         | 91                         | 00402                                     | CMPB                                | DBG\$GB_LANGUAGE, #8   | 7496         |
|       | 50 |                      | 08             | 38                         | 12                         | 00409                                     | BNEQ                                | 52\$   |              |
|       | 13 |                      | 16             | AC                         | DO                         | 0040B                                     | MOVL                                | LEFT_ARG, R0   | 7499         |
|       |    |                      |                | A0                         | 91                         | 0040F                                     | CMPB                                | 22(R0), #19  |              |
|       |    |                      |                | 0B                         | 12                         | 00413                                     | BNEQ                                | 51\$   |              |
| 0000V | CF |                      |                | 50                         | DD                         | 00415                                     | PUSHL                               | R0   | 7501         |
| 08    | AC |                      |                | 01                         | FB                         | 00417                                     | CALLS                               | #1, MAP NRO DTYPE_IN_RPG   |              |
|       | 1F |                      | 0C             | 50                         | DO                         | 0041C                                     | MOVL                                | R0, LEFT_ARG   |              |
|       | 32 |                      |                | AE                         | E9                         | 00420                                     | BLBC                                | BINARY_FLAG, 52\$  | 7503         |
|       | 33 |                      |                | 52                         | D1                         | 00424                                     | CMPL                                | OPCODE, #50  | 7506         |
|       |    |                      |                | 1A                         | 13                         | 00427                                     | BEQL                                | 52\$   |              |
|       |    |                      |                | 52                         | D1                         | 00429                                     | CMPL                                | OPCODE, #51  | 7507         |
|       | 50 |                      | 0C             | 15                         | 13                         | 0042C                                     | BEQL                                | 52\$   |              |
|       | 13 |                      | 16             | AC                         | DO                         | 0042E                                     | MOVL                                | RIGHT_ARG, R0  | 7509         |
|       |    |                      |                | A0                         | 91                         | 00432                                     | CMPB                                | 22(R0), #19  |              |
|       |    |                      |                | 0B                         | 12                         | 00436                                     | BNEQ                                | 52\$   |              |
|       |    |                      |                | 50                         | DD                         | 00438                                     | PUSHL                               | R0   | 7511         |
| 0000V | CF |                      |                | 01                         | FB                         | 0043A                                     | CALLS                               | #1, MAP NRO DTYPE_IN_RPG   |              |
| 0C    | AC |                      |                | 50                         | DO                         | 0043F                                     | MOVL                                | R0, RIGHT_ARG  |              |
|       | 50 |                      | 08             | AC                         | DO                         | 00443                                     | MOVL                                | LEFT_ARG, R0   | 7518         |
|       | 8F |                      | 03             | A0                         | 91                         | 00447                                     | CMPB                                | 3(R0), #255  |              |
|       |    |                      |                | 1C                         | 12                         | 0044C                                     | BNEQ                                | 53\$   |              |
|       | 8F |                      | 17             | A0                         | 91                         | 0044E                                     | CMPB                                | 23(R0), #255   | 7519         |
|       |    |                      |                | 15                         | 12                         | 00453                                     | BNEQ                                | 53\$   |              |
|       | 8F |                      | 16             | A0                         | 91                         | 00455                                     | CMPB                                | 22(R0), #255   | 7520         |
|       |    |                      | 0C             | 0E                         | 12                         | 0045A                                     | BNEQ                                | 53\$   |              |
|       |    |                      |                | AC                         | DD                         | 0045C                                     | PUSHL                               | RIGHT_ARG  | 7522         |
|       |    |                      |                | 50                         | DD                         | 0045F                                     | PUSHL                               | R0   |              |
| 0000V | CF |                      |                | 02                         | FB                         | 00461                                     | CALLS                               | #2, FIXUP_EMPTY_SET  |              |
| 08    | AC |                      | 0C             | 50                         | DO                         | 00466                                     | MOVL                                | R0, LEFT_ARG   |              |
|       | 27 |                      |                | AE                         | E9                         | 0046A                                     | BLBC                                | BINARY_FLAG, 54\$  | 7524         |
|       | 50 |                      | 0C             | AC                         | DO                         | 0046E                                     | MOVL                                | RIGHT_ARG, R0  | 7526         |
|       | 8F |                      | 03             | A0                         | 91                         | 00472                                     | CMPB                                | 3(R0), #255  |              |
|       |    |                      |                | 1C                         | 12                         | 00477                                     | BNEQ                                | 54\$   |              |
|       | 8F |                      | 17             | A0                         | 91                         | 00479                                     | CMPB                                | 23(R0), #255   | 7527         |
|       |    |                      |                | 15                         | 12                         | 0047E                                     | BNEQ                                | 54\$   |              |
|       | 8F |                      | 16             | A0                         | 91                         | 00480                                     | CMPB                                | 22(R0), #255   | 7528         |
|       |    |                      | 08             | 0E                         | 12                         | 00485                                     | BNEQ                                | 54\$   |              |
|       |    |                      |                | AC                         | DD                         | 00487                                     | PUSHL                               | LEFT_ARG   | 7530         |
|       |    |                      |                | 50                         | DD                         | 0048A                                     | PUSHL                               | R0   |              |
| 0000V | CF |                      |                | 02                         | FB                         | 0048C                                     | CALLS                               | #2, FIXUP_EMPTY_SET  |              |
| 0C    | AC |                      | 08             | 50                         | DO                         | 00491                                     | MOVL                                | R0, RIGHT_ARG  |              |
|       | 53 |                      |                | AC                         | DO                         | 00495                                     | MOVL                                | LEFT_ARG, R3   | 7536         |
|       |    |                      |                | 53                         | DD                         | 00499                                     | PUSHL                               | R3   |              |
| F962  | CF |                      | 0C             | 01                         | FB                         | 0049B                                     | CALLS                               | #1, DBG\$DO_MAPPING  |              |
|       | 16 |                      |                | AE                         | E9                         | 004A0                                     | BLBC                                | BINARY_FLAG, 56\$  | 7537         |
|       | 32 |                      |                | 52                         | D1                         | 004A4                                     | CMPL                                | OPCODE, #50  | 7539         |
|       |    |                      |                | 09                         | 12                         | 004A7                                     | BNEQ                                | 55\$   |              |

|            |    |            |      |      |       |       |       |                            |      |
|------------|----|------------|------|------|-------|-------|-------|----------------------------|------|
|            |    | 50         | OC   | AC   | DO    | 004A9 | MOVL  | RIGHT_ARG, R0              | 7540 |
|            |    |            | 04   | AO   | 95    | 004AD | TSTB  | 4(R0)                      |      |
|            |    |            |      | 08   | 19    | 004B0 | BLSS  | 56\$                       |      |
|            |    |            | OC   | AC   | DD    | 004B2 | PUSHL | RIGHT_ARG                  | 7542 |
| F948       | CF |            |      | 01   | FB    | 004B5 | CALLS | #1, DBG\$DO_MAPPING        |      |
|            |    |            |      | 53   | DD    | 004BA | PUSHL | R3                         | 7547 |
| 0000V      | CF |            |      | 01   | FB    | 004BC | CALLS | #1, DBG\$GET_DTYPE         |      |
|            | 56 |            |      | 50   | DO    | 004C1 | MOVL  | R0, LEFT_TYPE              |      |
| 18         | AE |            |      | 56   | DO    | 004C4 | MOVL  | LEFT_TYPE, NEW_LEFT_TYPE   | 7548 |
|            | 11 |            | OC   | AE   | E9    | 004C8 | BLBC  | BINARY_FLAG, 57\$          | 7553 |
|            |    |            | OC   | AC   | DD    | 004CC | PUSHL | RIGHT_ARG                  | 7555 |
| 0000V      | CF |            |      | 01   | FB    | 004CF | CALLS | #1, DBG\$GET_DTYPE         |      |
|            | 58 |            |      | 50   | DO    | 004D4 | MOVL  | R0, RIGHT_TYPE             |      |
| 14         | AE |            |      | 58   | DO    | 004D7 | MOVL  | RIGHT_TYPE, NEW_RIGHT_TYPE | 7556 |
|            |    |            |      | 05   | 11    | 004DB | BRB   | 58\$                       | 7553 |
|            |    |            |      | 58   | D4    | 004DD | CLRL  | RIGHT_TYPE                 | 7563 |
|            |    | 38         | 14   | AE   | D4    | 004DF | CLRL  | NEW_RIGHT_TYPE             | 7564 |
|            |    |            |      | 52   | D1    | 004E2 | CMPL  | OPCODE, #56                | 7576 |
|            |    |            |      | 19   | 12    | 004E5 | BNEQ  | 60\$                       |      |
| OF         |    |            |      | 05   | E1    | 004E7 | BBC   | #5, 4(R3), 59\$            | 7579 |
| 04         | A3 |            |      | 7E   | D4    | 004EC | CLRL  | -(SP)                      | 7581 |
|            |    |            |      | 53   | DD    | 004EE | PUSHL | R3                         |      |
|            |    |            |      | 53   | DD    | 004F0 | PUSHL | R3                         |      |
| F047       | CF |            |      | 03   | FB    | 004F2 | CALLS | #3, DBG\$CONV_TEXT_VALUE   |      |
| 08         | AC |            |      | 50   | DO    | 004F7 | MOVL  | R0, LEFT_ARG               |      |
|            | 50 |            | 08   | AC   | DO    | 004FB | MOVL  | LEFT_ARG, R0               | 7583 |
|            |    |            |      | 04   | 004FF | RET   |       |                            |      |
|            | 03 |            | OC   | AE   | E8    | 00500 | BLBS  | BINARY_FLAG, 61\$          | 7657 |
|            |    |            |      | 0278 | 31    | 00504 | BRW   | 90\$                       |      |
|            | 40 |            |      | 54   | E9    | 00507 | BLBC  | R4, 65\$                   | 7624 |
|            | 50 |            | OC   | AC   | DO    | 0050A | MOVL  | RIGHT_ARG, R0              | 7625 |
|            |    |            | 04   | AO   | 95    | 0050E | TSTB  | 4(R0)                      |      |
|            |    |            |      | 0A   | 19    | 00511 | BLSS  | 62\$                       |      |
|            | 0E |            |      | 56   | D1    | 00513 | CMPL  | LEFT_TYPE, #14             | 7626 |
|            |    |            |      | 2F   | 12    | 00516 | BNEQ  | 64\$                       |      |
|            | 16 |            |      | 58   | D1    | 00518 | CMPL  | RIGHT_TYPE, #22            |      |
|            |    |            |      | 2A   | 12    | 0051B | BNEQ  | 64\$                       |      |
|            | 50 |            | 08   | AC   | DO    | 0051D | MOVL  | LEFT_ARG, R0               | 7629 |
| OF         | 04 | AO         |      | 05   | E1    | 00521 | BBC   | #5, 4(R0), 63\$            |      |
|            |    |            | 0101 | 8F   | BB    | 00526 | PUSHR | #*H<R0,R8>                 | 7631 |
|            |    |            |      | 50   | DD    | 0052A | PUSHL | R0                         |      |
| FOOD       | CF |            |      | 03   | FB    | 0052C | CALLS | #3, DBG\$CONV_TEXT_VALUE   |      |
| 08         | AC |            |      | 50   | DO    | 00531 | MOVL  | R0, LEFT_ARG               |      |
|            |    | 000000000  |      | EF   | DD    | 00535 | PUSHL | CVT_ROUND_FLAG             | 7633 |
|            | 7E |            | 08   | AC   | 7D    | 0053B | MOVQ  | LEFT_ARG, -(SP)            |      |
| 000000000G | 00 |            |      | 03   | FB    | 0053F | CALLS | #3, DBG\$COVER_DX_DX       |      |
|            |    |            |      | 04   | 00546 | RET   |       |                            |      |
|            | 05 |            |      | 54   | E8    | 00547 | BLBS  | R4, 66\$                   | 7642 |
|            | 33 |            |      | 52   | D1    | 0054A | CMPL  | OPCODE, #51                | 7643 |
|            |    |            |      | 21   | 12    | 0054D | BNEQ  | 67\$                       |      |
|            | 50 |            | OC   | AC   | DO    | 0054F | MOVL  | RIGHT_ARG, R0              | 7646 |
|            | 05 |            | 06   | AO   | 91    | 00553 | CMPB  | 6(R0), #5                  |      |
|            |    |            |      | 17   | 12    | 00557 | BNEQ  | 67\$                       |      |
|            | 50 |            | 08   | AC   | DO    | 00559 | MOVL  | LEFT_ARG, R0               | 7647 |
|            | 0E |            | 16   | AO   | 91    | 0055D | CMPB  | 22(R0), #14                |      |
|            |    |            |      | 0D   | 13    | 00561 | BEQL  | 67\$                       |      |
|            | 03 | 000000000G |      | 00   | 91    | 00563 | CMPB  | DBG\$GB_LANGUAGE, #3       | 7648 |

|           |    |           |           |       |       |                       |                      |                      |
|-----------|----|-----------|-----------|-------|-------|-----------------------|----------------------|----------------------|
| 14        | AE | 04        | 12        | 0056A | BNEQ  | 67\$                  |                      |                      |
|           | 08 | 30        | D0        | 0056C | MOVL  | #48, NEW_RIGHT_TYPE   |                      | 7650                 |
|           | 33 | 54        | E8        | 00570 | BLBS  | R4, 68\$              |                      | 7654                 |
|           |    | 52        | D1        | 00573 | CMPL  | OPCODE, #51           |                      |                      |
|           |    | 03        | 13        | 00576 | BEQL  | 68\$                  |                      |                      |
|           |    | 01A0      | 31        | 00578 | BRW   | 86\$                  |                      |                      |
|           |    | 04        | AE        | DD    | 0057B | 68\$: PUSHL           | ROUT_TBL_SIZE        | 7659                 |
|           |    | 57        | DD        | 0057E | PUSHL | ROUT_TBL              |                      |                      |
|           |    | 10        | AE        | DD    | 00580 | PUSHL                 | INCOMP_TBL_SIZE      | 7658                 |
|           |    | 0220      | 8F        | BB    | 00583 | PUSHR                 | #*M<R5,R9>           | 7657                 |
|           |    | 5A        | DD        | 00587 | PUSHL | HIER_TBL              |                      |                      |
|           |    | 7E        | D4        | 00589 | CLRL  | -(SPT                 |                      | 7656                 |
|           |    | 30        | AE        | DD    | 0058B | PUSHL                 | NEW_RIGHT_TYPE       |                      |
|           |    | 38        | AE        | DD    | 0058E | PUSHL                 | NEW_LEFT_TYPE        |                      |
| 0000V     | CF | 09        | FB        | 00591 | CALLS | #9, FIND_PATH_DEPOSIT |                      |                      |
|           | 03 | 50        | E8        | 00596 | BLBS  | R0, 69\$              |                      |                      |
|           |    | 0208      | 31        | 00599 | BRW   | 92\$                  |                      |                      |
|           | 53 | 0C        | AC        | D0    | 0059C | 69\$: MOVL            | RIGHT_ARG, R3        | 7667                 |
|           | 05 | 06        | A3        | 91    | 005A0 | CMPB                  | 6(R3), #5            |                      |
|           |    | 17        | 12        | 005A4 | BNEQ  | 70\$                  |                      |                      |
|           | 50 | 08        | AC        | D0    | 005A6 | MOVL                  | LEFT_ARG, R0         | 7668                 |
|           | 0E | 16        | A0        | 91    | 005AA | CMPB                  | 22(R0), #14          |                      |
|           |    |           | 0D        | 13    | 005AE | BEQL                  | 70\$                 |                      |
|           | 03 | 00000000G | 00        | 91    | 005B0 | CMPB                  | DBG\$GB_LANGUAGE, #3 | 7669                 |
|           |    |           | 04        | 12    | 005B7 | BNEQ                  | 70\$                 |                      |
| 16        | A3 |           | 30        | 90    | 005B9 | MOVB                  | #48, 22(R3)          | 7671                 |
|           | 54 | 14        | A3        | 9E    | 005BD | 70\$: MOVAB           | 20(R3), R4           | 7676                 |
|           |    |           | 6E        | D4    | 005C1 | CLRL                  | (SP)                 |                      |
|           | 09 | 03        | A4        | 91    | 005C3 | CMPB                  | 3(R4), #9            |                      |
|           |    |           | 0B        | 12    | 005C7 | BNEQ                  | 71\$                 |                      |
|           |    |           | 6E        | D6    | 005C9 | INCL                  | (SP)                 |                      |
| 04        | 1E | A3        | 03        | E0    | 005CB | BBS                   | #3, 30(R3), 71\$     | 7677                 |
|           | 14 | AE        | 15        | D0    | 005D0 | MOVL                  | #21, NEW_RIGHT_TYPE  | 7679                 |
|           |    | 52        | 08        | AC    | D0    | 005D4                 | 71\$: MOVL           | LEFT_ARG, R2         |
| 56        | 04 | A2        | 05        | E1    | 005D8 | BBC                   | #5, 2(R2), 75\$      | 7686                 |
|           |    | 50        | 00000000G | 00    | 9A    | 005DD                 | MOVZBL               | DBG\$GB_LANGUAGE, R0 |
|           |    | 03        | 50        | 91    | 005E4 | CMPB                  | R0, #3               | 7696                 |
|           |    |           | 05        | 13    | 005E7 | BEQL                  | 72\$                 |                      |
|           | 08 |           | 50        | 91    | 005E9 | CMPB                  | R0, #8               | 7697                 |
|           |    |           | 36        | 12    | 005EC | BNEQ                  | 74\$                 |                      |
|           | 03 | 02        | A4        | 91    | 005EE | 72\$: CMPB            | 2(R4), #3            | 7704                 |
|           |    |           | 12        | 13    | 005F2 | BEQL                  | 73\$                 |                      |
|           | 04 | 02        | A4        | 91    | 005F4 | CMPB                  | 2(R4), #4            | 7705                 |
|           |    |           | 0C        | 13    | 005F8 | BEQL                  | 73\$                 |                      |
|           | 05 | 02        | A4        | 91    | 005FA | CMPB                  | 2(R4), #5            | 7706                 |
|           |    |           | 06        | 13    | 005FE | BEQL                  | 73\$                 |                      |
|           | 0F | 02        | A4        | 91    | 00600 | CMPB                  | 2(R4), #15           | 7707                 |
|           |    |           | 1E        | 12    | 00604 | BNEQ                  | 74\$                 |                      |
| 0042      | 8F | 12        | A2        | B1    | 00606 | 73\$: CMPW            | 18(R2), #66          | 7710                 |
|           |    |           | 16        | 12    | 0060C | BNEQ                  | 74\$                 |                      |
| 12        | A2 | 43        | 8F        | 9B    | 0060E | MOVZBW                | #67, 18(R2)          | 7713                 |
|           |    |           | 5B        | DD    | 00613 | PUSHL                 | R11                  | 7714                 |
|           |    |           | 01        | DD    | 00615 | PUSHL                 | #1                   |                      |
|           |    | 0002874B  | 8F        | DD    | 00617 | PUSHL                 | #165707              |                      |
| 00000000G | 00 |           | 03        | FB    | 0061D | CALLS                 | #3, LIB\$SIGNAL      |                      |
|           |    | 0104      | 8F        | BB    | 00624 | 74\$: PUSHR           | #*M<R2,R8>           | 7721                 |
|           |    |           | 52        | DD    | 00628 | PUSHL                 | R2                   |                      |

|    |             |          |           |          |                |               |  |                 |
|----|-------------|----------|-----------|----------|----------------|---------------|--|-----------------|
| 12 | EFOF<br>08  | CF<br>AC | 03<br>50  | FB<br>DO | 0062A<br>0062F | CALLS<br>MOVL | #3, DBG\$CONV_TEXT_VALUE<br>R0, LEFT_ARG | 7730            |
|    | 1E          | 05<br>A3 | 6E<br>03  | E9<br>E1 | 00633<br>00636 | 75%:<br>BLBC  | (SP), 76%                                | 7731            |
|    |             | 15       | A4        | 91       | 0063B          | 76%:<br>CMPB  | #3, 30(R3), 77%                          | 7732            |
|    |             |          | 0C        | 13       | 0063F          | BEQL          | 2(R4), #21                               |                 |
|    |             | 0F       | A4        | 91       | 00641          | CMPB          | 77%                                      | 7733            |
|    |             |          | 06        | 13       | 00645          | BEQL          | 2(R4), #15                               |                 |
|    |             | 30       | A4        | 91       | 00647          | CMPB          | 77%                                      | 7734            |
|    |             |          | 0F        | 12       | 0064B          | BNEQ          | 2(R4), #48                               |                 |
|    |             | 7E       | 15        | 7D       | 0064D          | 77%:<br>MOVQ  | 78%                                      | 7736            |
|    |             |          | AC        | DD       | 00650          | PUSHL         | #21, -(SP)                               |                 |
|    | 0000V<br>08 | CF       | 03        | FB       | 00653          | CALLS         | LEFT_ARG                                 |                 |
|    |             | AC       | 50        | DO       | 0065B          | MOVL          | #3, INTMED_DATA_FOR_DEP                  |                 |
|    |             | 50       | 00000000G | 00       | 9A             | 0065C         | R0, LEFT_ARG                             | 7750            |
|    |             | 03       | 50        | 91       | 00663          | MOVZBL        | DBG\$GB_LANGUAGE, R0                     |                 |
|    |             |          | 05        | 13       | 00666          | CMPB          | R0, #3                                   |                 |
|    |             | 08       | 50        | 91       | 00668          | BEQL          | 79%                                      | 7751            |
|    |             |          | 50        | 12       | 0066B          | CMPB          | R0, #8                                   |                 |
|    |             | 03       | A4        | 91       | 0066D          | 79%:<br>BNEQ  | 81%                                      | 7758            |
|    |             |          | 12        | 13       | 00671          | CMPB          | 2(R4), #3                                |                 |
|    |             | 04       | A4        | 91       | 00673          | BEQL          | 80%                                      | 7759            |
|    |             |          | 0C        | 13       | 00677          | CMPB          | 2(R4), #4                                |                 |
|    |             | 05       | A4        | 91       | 00679          | BEQL          | 80%                                      | 7760            |
|    |             |          | 06        | 13       | 0067D          | CMPB          | 2(R4), #5                                |                 |
|    |             | 0F       | A4        | 91       | 0067F          | BEQL          | 80%                                      | 7761            |
|    |             |          | 38        | 12       | 00683          | CMPB          | 2(R4), #15                               |                 |
|    |             | 50       | AC        | DO       | 00685          | 80%:<br>BNEQ  | 81%                                      | 7762            |
|    |             | 15       | A0        | 91       | 00689          | MOVL          | LEFT_ARG, R0                             |                 |
|    |             |          | 2E        | 12       | 0068D          | CMPB          | 22(R0), #21                              |                 |
|    |             | 51       | AC        | DO       | 0068F          | BNEQ          | 81%                                      | 7770            |
|    |             | 50       | A1        | 3C       | 00693          | MOVL          | LEFT_ARG, R1                             |                 |
|    |             | 50       | 02        | C6       | 00697          | MOVZWL        | 20(RT), DIG                              | 7772            |
|    |             | 50       | A1        | C0       | 0069A          | DIVL2         | #2, R0                                   |                 |
| 51 |             | 60       | F2        | 8F       | 8B             | 0069E         | ADDL2                                    | 24(R1), R0      |
|    |             | 0D       | 51        | 91       | 006A3          | BICB3         | #-14, (SIGN_POS), SIGN                   | 7777            |
|    |             |          | 15        | 12       | 006A6          | CMPB          | SIGN, #13                                | 7778            |
|    |             | 60       | 8F        | 8A       | 006A8          | BNEQ          | 81%                                      | 7781            |
|    |             |          | 5B        | DD       | 006AC          | BICB2         | #-255, (SIGN_POS)                        | 7782            |
|    |             |          | 01        | DD       | 006AE          | PUSHL         | R11                                      |                 |
|    |             | 0002874B | 8F        | DD       | 006B0          | PUSHL         | #1                                       |                 |
|    | 00000000G   | 00       | 03        | FB       | 006B6          | PUSHL         | #165707                                  |                 |
|    |             | 50       | 00000000G | 00       | 9A             | 006BD         | 81%:<br>CALLS                            | #3, LIB\$SIGNAL |
|    |             | 03       | 50        | 91       | 006C4          | MOVZBL        | DBG\$GB_LANGUAGE, R0                     | 7806            |
|    |             |          | 05        | 13       | 006C7          | CMPB          | R0, #3                                   |                 |
|    |             | 08       | 50        | 91       | 006C9          | BEQL          | 82%                                      | 7807            |
|    |             |          | 24        | 12       | 006CC          | CMPB          | R0, #8                                   |                 |
|    |             | 21       | 6E        | E9       | 006CE          | 82%:<br>BNEQ  | 83%                                      | 7810            |
|    |             |          | 01        | DD       | 006D1          | BLBC          | (SP), 83%                                | 7821            |
|    |             |          | 15        | DD       | 006D3          | PUSHL         | #1                                       |                 |
|    |             |          | 53        | DD       | 006D5          | PUSHL         | #21                                      |                 |
|    | 0000V       | CF       | 03        | FB       | 006D7          | PUSHL         | R3                                       |                 |
|    |             | 52       | 50        | DO       | 006DC          | CALLS         | #3, INTMED_DATA_FOR_DEP                  |                 |
|    |             | 53       | 52        | D1       | 006DF          | MOVL          | R0, NEW_RIGHT_ARG                        | 7829            |
|    |             |          | 0E        | 13       | 006E2          | CMPL          | NEW_RIGHT_ARG, R3                        |                 |
|    |             |          | 52        | DD       | 006E4          | BEQL          | 83%                                      | 7832            |
|    |             |          | 08        | AC       | DD             | 006E6         | PUSHL                                    | NEW_RIGHT_ARG   |
|    |             |          |           |          |                | PUSHL         | LEFT_ARG                                 |                 |

|       |    |           |    |    |       |        |                      |      |
|-------|----|-----------|----|----|-------|--------|----------------------|------|
| 0000V | CF |           | 02 | FB | 006E9 | CALLS  | #2, DBGSTYPE_CONV    |      |
| 08    | AC |           | 50 | DD | 006EE | MOVL   | R0, LEFT_ARG         |      |
|       |    | 08        | 53 | DD | 006F2 | PUSHL  | R3                   | 7837 |
|       |    |           | AC | DD | 006F4 | PUSHL  | LEFT_ARG             |      |
| 0000V | CF |           | 02 | FB | 006F7 | CALLS  | #2, DBGSTYPE_CONV    |      |
| 0C    | AC |           | 50 | DD | 006FC | MOVL   | R0, RIGHT_ARG        |      |
|       | 05 | 06        | A0 | 91 | 00700 | CMPB   | 6(R0), #5            | 7839 |
|       |    |           | 0D | 12 | 00704 | BNEQ   | 84\$                 |      |
|       | 30 | 16        | A0 | 91 | 00706 | CMPB   | 22(R0), #48          | 7840 |
|       |    |           | 07 | 12 | 0070A | BNEQ   | 84\$                 |      |
|       | 03 | 00000000G | 00 | 91 | 0070C | CMPB   | DBG\$GB_LANGUAGE, #3 | 7841 |
|       |    |           | 01 | 13 | 00713 | BEQL   | 85\$                 |      |
|       |    |           |    | 04 | 00715 | RET    |                      |      |
| 16    | A0 |           | 0E | 90 | 00716 | MOVB   | #14, 22(R0)          | 7843 |
|       |    |           |    | 04 | 0071A | RET    |                      | 7845 |
|       | 04 | 00000000G | 00 | 91 | 0071B | CMPB   | DBG\$GB_LANGUAGE, #4 | 7853 |
|       |    |           | 2F | 12 | 00722 | BNEQ   | 89\$                 |      |
|       | 15 | 18        | AE | D1 | 00724 | CMPL   | NEW_LEFT_TYPE, #21   | 7856 |
|       |    |           | 10 | 12 | 00728 | BNEQ   | 87\$                 |      |
|       |    | 14        | AE | DD | 0072A | PUSHL  | NEW_RIGHT_TYPE       | 7858 |
|       | 50 | 08        | AC | DD | 0072D | MOVL   | LEFT_ARG, R0         |      |
|       | 7E | 14        | A0 | 3C | 00731 | MOVZWL | 20(R0), -(SP)        |      |
|       |    | 20        | AE | 9F | 00735 | PUSHAB | NEW_LEFT_TYPE        |      |
|       |    |           | 14 | 11 | 00738 | BRB    | 88\$                 |      |
|       | 15 | 14        | AE | D1 | 0073A | CMPL   | NEW_RIGHT_TYPE, #21  | 7860 |
|       |    |           | 13 | 12 | 0073E | BNEQ   | 89\$                 |      |
|       |    | 18        | AE | DD | 00740 | PUSHL  | NEW_LEFT_TYPE        | 7862 |
|       | 50 | 0C        | AC | DD | 00743 | MOVL   | RIGHT_ARG, R0        |      |
|       | 7E | 14        | A0 | 3C | 00747 | MOVZWL | 20(R0), -(SP)        |      |
|       |    | 1C        | AE | 9F | 0074B | PUSHAB | NEW_RIGHT_TYPE       |      |
| 0000V | CF |           | 03 | FB | 0074E | CALLS  | #3, MAP_PACKED       |      |
|       |    | 04        | AE | DD | 00753 | PUSHL  | ROUT_TBL_SIZE        | 7873 |
|       |    |           | 57 | DD | 00756 | PUSHL  | ROUT_TBL             |      |
|       |    | 10        | AE | DD | 00758 | PUSHL  | INCOMP_TBL_SIZE      | 7872 |
|       |    | 0220      | 8F | BB | 0075B | PUSHR  | #*M<R5,R9>           | 7871 |
|       |    |           | 5A | DD | 0075F | PUSHL  | HIER_TBL             |      |
|       |    | 00000000' | EF | DD | 00761 | PUSHL  | MAX_DEPTH            | 7870 |
|       |    |           | 7E | D4 | 00767 | CLRL   | -(SP)                | 7866 |
|       |    | 30        | AE | 9F | 00769 | PUSHAB | ROUT_TBL_INDEX       |      |
|       |    | 38        | AE | 9F | 0076C | PUSHAB | NEW_RIGHT_TYPE       |      |
|       |    | 40        | AE | 9F | 0076F | PUSHAB | NEW_LEFT_TYPE        |      |
|       |    | 40        | AE | DD | 00772 | PUSHL  | NEW_RIGHT_TYPE       | 7867 |
|       |    | 48        | AE | DD | 00775 | PUSHL  | NEW_LEFT_TYPE        |      |
| 0000V | CF |           | 0D | FB | 00778 | CALLS  | #13, FIND_JOIN       |      |
|       |    |           | 1C | 11 | 0077D | BRB    | 91\$                 | 7866 |
|       |    | 04        | AE | DD | 0077F | PUSHL  | ROUT_TBL_SIZE        | 7888 |
|       |    |           | 57 | DD | 00782 | PUSHL  | ROUT_TBL             |      |
|       |    | 10        | AE | DD | 00784 | PUSHL  | INCOMP_TBL_SIZE      | 7887 |
|       |    | 0220      | 8F | BB | 00787 | PUSHR  | #*M<R5,R9>           | 7886 |
|       |    |           | 5A | DD | 0078B | PUSHL  | HIER_TBL             |      |
|       |    | 28        | AE | 9F | 0078D | PUSHAB | ROUT_TBL_INDEX       | 7883 |
|       |    | 34        | AE | 9F | 00790 | PUSHAB | NEW_LEFT_TYPE        |      |
|       |    | 38        | AE | DD | 00793 | PUSHL  | NEW_LEFT_TYPE        | 7884 |
| 0000V | CF |           | 09 | FB | 00796 | CALLS  | #9, FIND_PATH        |      |
|       | 50 | 00000000' | EF | D1 | 0079B | CMPL   | MAX_DEPTH, R0        | 7883 |
|       |    |           | 11 | 12 | 007A2 | BNEQ   | 93\$                 |      |
|       |    |           | 5B | DD | 007A4 | PUSHL  | R11                  | 7890 |

|           |    |           |      |    |       |        |                             |                      |      |
|-----------|----|-----------|------|----|-------|--------|-----------------------------|----------------------|------|
|           |    |           | 01   | DD | 007A6 | PUSHL  | #1                          |                      |      |
|           |    |           | 8F   | DD | 007A8 | PUSHL  | #166346                     |                      |      |
| 00000000G | 00 | 000289CA  | 03   | FB | 007AE | CALLS  | #3, LIB\$SIGNAL             |                      |      |
|           | 2C | 0C        | AE   | E9 | 007B5 | BLBC   | BINARY_FLAG, 94\$           |                      | 7902 |
|           | 05 | 00000000G | 00   | 91 | 007B9 | CMPB   | DBG\$GB_LANGUAGE, #5        |                      | 7903 |
|           |    |           | 23   | 12 | 007C0 | BNEQ   | 94\$                        |                      |      |
|           |    | 04        | AE   | DD | 007C2 | PUSHL  | ROUT_TBL_SIZE               |                      | 7909 |
|           |    |           | 57   | DD | 007C5 | PUSHL  | ROUT_TBL                    |                      |      |
|           |    | 10        | AE   | DD | 007C7 | PUSHL  | INCOMP_TBL_SIZE             |                      | 7908 |
|           |    | 0220      | 8F   | BB | 007CA | PUSHR  | #*M<R5,R9>                  |                      | 7907 |
|           |    |           | 5A   | DD | 007CE | PUSHL  | HIER_TBL                    |                      |      |
|           |    | 28        | AE   | 9F | 007D0 | PUSHAB | ROUT_TBL_INDEX              |                      | 7905 |
|           |    | 30        | AE   | 9F | 007D3 | PUSHAB | NEW_RIGHT_TYPE              |                      |      |
|           |    | 38        | AE   | 9F | 007D6 | PUSHAB | NEW_LEFT_TYPE               |                      |      |
|           |    | 0140      | 8F   | BB | 007D9 | PUSHR  | #*M<R6,R8>                  |                      |      |
|           |    | 04        | AC   | DD | 007DD | PUSHL  | OPERATOR                    |                      |      |
| 0000V     | CF |           | 0C   | FB | 007E0 | CALLS  | #12, MODIFY_PLI_TARGET_TYPE |                      |      |
|           | 50 |           | 08   | AC | D0    | 007E5  | MOVL                        | LEFT_ARG, R0         | 7914 |
| 10        | 04 | AO        | 05   | E1 | 007E9 | BBC    | #5, Z(R0) 95\$              |                      |      |
|           |    |           | 18   | AE | DD    | 007EE  | PUSHL                       | NEW_LEFT_TYPE        | 7916 |
|           |    |           | 50   | DD | 007F1 | PUSHL  | R0                          |                      |      |
|           |    |           | 50   | DD | 007F3 | PUSHL  | R0                          |                      |      |
| ED44      | CF |           | 03   | FB | 007F5 | CALLS  | #3, DBG\$CONV_TEXT_VALUE    |                      |      |
| 08        | AC |           | 50   | D0 | 007FA | MOVL   | R0, LEFT_ARG                |                      |      |
|           | 53 |           | 08   | AC | D0    | 007FE  | MOVL                        | LEFT_ARG, R3         | 7927 |
| 18        | AE |           | 56   | D1 | 00802 | CMPB   | LEFT_TYPE, NEW_LEFT_TYPE    |                      | 7923 |
|           |    |           | 03   | 12 | 00806 | BNEQ   | 96\$                        |                      |      |
|           |    |           | 0089 | 31 | 00808 | BRW    | 102\$                       |                      |      |
|           | 7E |           | 14   | A3 | 3C    | 0080B  | MOVZWL                      | 20(R3), -(SP)        | 7927 |
|           |    |           | 1C   | AE | DD    | 0080F  | PUSHL                       | NEW_LEFT_TYPE        | 7926 |
|           |    |           | 56   | DD | 00812 | PUSHL  | LEFT_TYPE                   |                      |      |
| 0000V     | CF |           | 03   | FB | 00814 | CALLS  | #3, GET_DATA_LENGTH         |                      |      |
|           | 55 |           | 50   | D0 | 00819 | MOVL   | R0, LENGTH                  |                      |      |
|           |    |           | 59   | D4 | 0081C | CLRL   | SD_CLASS_FLAG               |                      | 7928 |
|           | 50 | 00000000G | 00   | 9A | 0081E | MOVZBL | DBG\$GB_LANGUAGE, R0        |                      | 7929 |
|           | 03 |           | 50   | 91 | 00825 | CMPB   | R0, #3                      |                      |      |
|           |    |           | 05   | 13 | 00828 | BEQL   | 97\$                        |                      |      |
|           | 08 |           | 50   | 91 | 0082A | CMPB   | R0, #8                      |                      | 7930 |
|           |    |           | 09   | 12 | 0082D | BNEQ   | 98\$                        |                      |      |
|           | 09 | 17        | A3   | 91 | 0082F | CMPB   | 23(R3), #9                  |                      | 7933 |
|           |    |           | 03   | 12 | 00833 | BNEQ   | 98\$                        |                      |      |
|           | 59 |           | 01   | D0 | 00835 | MOVL   | #1, SD_CLASS_FLAG           |                      | 7935 |
|           |    | 0208      | 8F   | BB | 00838 | PUSHR  | #*M<R3,R9>                  |                      | 7940 |
|           |    |           | 55   | DD | 0083C | PUSHL  | LENGTH                      |                      | 7939 |
|           |    |           | 24   | AE | DD    | 0083E  | PUSHL                       | NEW_LEFT_TYPE        | 7938 |
| 0000V     | CF |           | 04   | FB | 00841 | CALLS  | #4, MAKE_VAL_DESC           |                      |      |
|           | 5A |           | 50   | D0 | 00846 | MOVL   | R0, NEW_LEFT_ARG            |                      |      |
|           | 09 |           | 17   | AA | 91    | 00849  | CMPB                        | 23(NEW_LEFT_ARG), #9 | 7943 |
|           |    |           | 37   | 12 | 0084D | BNEQ   | 101\$                       |                      |      |
|           | 0A |           | 16   | A3 | 91    | 0084F  | CMPB                        | 22(R3), #10          | 7952 |
|           |    |           | 12   | 13 | 00853 | BEQL   | 99\$                        |                      |      |
|           | 0B |           | 16   | A3 | 91    | 00855  | CMPB                        | 22(R3), #11          | 7953 |
|           |    |           | 0C   | 13 | 00859 | BEQL   | 99\$                        |                      |      |
|           | 1B |           | 16   | A3 | 91    | 0085B  | CMPB                        | 22(R3), #27          | 7954 |
|           |    |           | 06   | 13 | 0085F | BEQL   | 99\$                        |                      |      |
|           | 1C |           | 16   | A3 | 91    | 00861  | CMPB                        | 22(R3), #28          | 7955 |
|           |    |           | 1A   | 12 | 00865 | BNEQ   | 100\$                       |                      |      |

|       |    |           |      |    |       |       |        |                            |      |
|-------|----|-----------|------|----|-------|-------|--------|----------------------------|------|
|       |    | 1C        | AE   | 9F | 00867 | 99%:  | PUSHAB | DIGITS                     | 7958 |
|       |    |           | 53   | DD | 0086A |       | PUSHL  | R3                         |      |
| 0000V | CF |           | 02   | FB | 0086C |       | CALLS  | #2, GET_SCALE              |      |
| 1C    | AA |           | 50   | 90 | 00871 |       | MOVB   | R0, 28(NEW_LEFT_ARG)       |      |
| 1D    | AA | 1C        | AE   | 90 | 00875 |       | MOVB   | DIGITS, 29(NEW_LEFT_ARG)   | 7959 |
| 14    | AA | 1C        | AE   | 80 | 0087A |       | MOVW   | DIGITS, 20(NEW_LEFT_ARG)   | 7960 |
|       |    |           | 05   | 11 | 0087F |       | BRB    | 101\$                      | 7952 |
| 1C    | AA | 1C        | A3   | 90 | 00881 | 100%: | MOVB   | 28(R3), 28(NEW_LEFT_ARG)   | 7964 |
|       |    | 0408      | 8F   | BB | 00886 | 101%: | PUSHR  | #*M<R3,R10>                | 7967 |
| 0000V | CF |           | 02   | FB | 0088A |       | CALLS  | #2, DBG\$TYPE_CONV         |      |
|       | 5A |           | 50   | D0 | 0088F |       | MOVL   | R0, NEW_LEFT_ARG           |      |
|       |    |           | 03   | 11 | 00892 |       | BRB    | 103\$                      | 7923 |
|       | 5A |           | 53   | D0 | 00894 | 102%: | MOVL   | R3, NEW_LEFT_ARG           | 7975 |
|       | 19 | OC        | AE   | E9 | 00897 | 103%: | BLBC   | BINARY_FLAG, -104\$        | 7980 |
|       | 50 | OC        | AC   | D0 | 00898 |       | MOVL   | RIGHT_ARG, R0              | 7982 |
| 10    | 04 |           | 05   | E1 | 0089F |       | BBC    | #5, 4(R0), 104\$           |      |
|       |    | 14        | AE   | DD | 008A4 |       | PUSHL  | NEW_RIGHT_TYPE             | 7984 |
|       |    |           | 50   | DD | 008A7 |       | PUSHL  | R0                         |      |
|       |    |           | 50   | DD | 008A9 |       | PUSHL  | R0                         |      |
| EC8E  | CF |           | 03   | FB | 008AB |       | CALLS  | #3, DBG\$CONV_TEXT_VALUE   |      |
| OC    | AC |           | 50   | D0 | 008B0 |       | MOVL   | R0, RIGHT_ARG              |      |
|       | 03 | OC        | AE   | E8 | 008B4 | 104%: | BLBS   | BINARY_FLAG, 105\$         | 7989 |
|       |    |           | 0099 | 31 | 008B8 |       | BRW    | 113\$                      |      |
|       | 53 | OC        | AC   | D0 | 008BB | 105%: | MOVL   | RIGHT_ARG, R3              | 7995 |
| 14    | AE |           | 58   | D1 | 008BF |       | CMPL   | RIGHT_TYPE, NEW_RIGHT_TYPE | 7991 |
|       |    |           | 03   | 12 | 008C3 |       | BNEQ   | 106\$                      |      |
|       |    |           | 0089 | 31 | 008C5 |       | BRW    | 112\$                      |      |
|       | 7E | 14        | A3   | 3C | 008C8 | 106%: | MOVZWL | 20(R3), -(SP)              | 7995 |
|       |    | 18        | AE   | DD | 008CC |       | PUSHL  | NEW_RIGHT_TYPE             | 7994 |
|       |    |           | 58   | DD | 008CF |       | PUSHL  | RIGHT_TYPE                 |      |
| 0000V | CF |           | 03   | FB | 008D1 |       | CALLS  | #3, GET_DATA_LENGTH        |      |
|       | 55 |           | 50   | D0 | 008D6 |       | MOVL   | R0, LENGTH                 |      |
|       |    |           | 59   | D4 | 008D9 |       | CLRL   | SD_CLASS_FLAG              | 7996 |
|       | 50 | 00000000G | 00   | 9A | 008DB |       | MOVZBL | DBG\$GB_LANGUAGE, R0       | 7997 |
|       | 03 |           | 50   | 91 | 008E2 |       | CPMB   | R0, #3                     |      |
|       |    |           | 05   | 13 | 008E5 |       | BEQL   | 107\$                      |      |
|       | 08 |           | 50   | 91 | 008E7 |       | CPMB   | R0, #8                     | 7998 |
|       |    |           | 09   | 12 | 008EA |       | BNEQ   | 108\$                      |      |
|       | 09 | 17        | A3   | 91 | 008EC | 107%: | CPMB   | 23(R3), #9                 | 8001 |
|       |    |           | 03   | 12 | 008F0 |       | BNEQ   | 108\$                      |      |
|       | 59 |           | 01   | D0 | 008F2 |       | MOVL   | #1, SD_CLASS_FLAG          | 8003 |
|       |    | 0208      | 8F   | BB | 008F5 | 108%: | PUSHR  | #*M<R3,R9>                 | 8008 |
|       |    |           | 55   | DD | 008F9 |       | PUSHL  | LENGTH                     | 8007 |
|       |    | 20        | AE   | DD | 008FB |       | PUSHL  | NEW_RIGHT_TYPE             | 8006 |
| 0000V | CF |           | 04   | FB | 008FE |       | CALLS  | #4, MAKE_VAL_DESC          |      |
|       | 52 |           | 50   | D0 | 00903 |       | MOVL   | R0, NEW_RIGHT_ARG          |      |
|       | 09 | 17        | A2   | 91 | 00906 |       | CPMB   | 23(NEW_RIGHT_ARG), #9      | 8011 |
|       |    |           | 37   | 12 | 0090A |       | BNEQ   | 111\$                      |      |
|       | 0A | 16        | A3   | 91 | 0090C |       | CPMB   | 22(R3), #10                | 8019 |
|       |    |           | 12   | 13 | 00910 |       | BEQL   | 109\$                      |      |
|       | 0B | 16        | A3   | 91 | 00912 |       | CPMB   | 22(R3), #11                | 8020 |
|       |    |           | 0C   | 13 | 00916 |       | BEQL   | 109\$                      |      |
|       | 1B | 16        | A3   | 91 | 00918 |       | CPMB   | 22(R3), #27                | 8021 |
|       |    |           | 06   | 13 | 0091C |       | BEQL   | 109\$                      |      |
|       | 1C | 16        | A3   | 91 | 0091E |       | CPMB   | 22(R3), #28                | 8022 |
|       |    |           | 1A   | 12 | 00922 |       | BNEQ   | 110\$                      |      |
|       |    | 1C        | AE   | 9F | 00924 | 109%: | PUSHAB | DIGITS                     | 8025 |

|          |    |    |      |    |       |        |                           |                                     |      |
|----------|----|----|------|----|-------|--------|---------------------------|-------------------------------------|------|
| 0000V    | CF |    | 53   | DD | 00927 | PUSHL  | R3                        |                                     |      |
| 1C       | A2 |    | 02   | FB | 00929 | CALLS  | #2, GET_SCALE             |                                     |      |
| 1D       | A2 | 1C | 50   | 90 | 0092E | MOVB   | R0, 28(NEW_RIGHT_ARG)     |                                     | 8026 |
| 14       | A2 | 1C | AE   | 90 | 00932 | MOVB   | DIGITS, 29(NEW_RIGHT_ARG) |                                     | 8027 |
|          |    |    | AE   | 80 | 00937 | MOVW   | DIGITS, 20(NEW_RIGHT_ARG) |                                     | 8019 |
|          |    |    | 05   | 11 | 0093C | BRB    | 111\$                     |                                     | 8031 |
| 1C       | A2 | 1C | A3   | 90 | 0093E | 110\$: | MOVB                      | 28(R3), 28(NEW_RIGHT_ARG)           | 8034 |
|          |    |    | 52   | DD | 00943 | 111\$: | PUSHL                     | NEW_RIGHT_ARG                       |      |
|          |    |    | 53   | DD | 00945 |        | PUSHL                     | R3                                  |      |
| 0000V    | CF |    | 02   | FB | 00947 | CALLS  | #2, DBG\$TYPE_CONV        |                                     | 7991 |
| 52       |    |    | 50   | DD | 0094C | MOVL   | R0, NEW_RIGHT_ARG         |                                     | 8038 |
|          |    |    | 03   | 11 | 0094F | BRB    | 113\$                     |                                     | 8041 |
| 52       |    |    | 53   | DD | 00951 | 112\$: | MOVL                      | R3, NEW_RIGHT_ARG                   |      |
| 50       |    | 10 | AE   | DD | 00954 | 113\$: | MOVL                      | ROUT_TBL_INDEX, R0                  |      |
|          |    | 02 | A740 | 7F | 00958 |        | PUSHAQ                    | 2(ROUT_TBL)[R0]                     |      |
| 53       |    |    | 9E   | 3C | 0095C |        | MOVZWL                    | 2(SP)+, ROUT_INDEX                  | 8042 |
|          |    | 04 | A740 | 7F | 0095F |        | PUSHAQ                    | 4(ROUT_TBL)[R0]                     |      |
| 54       |    |    | 9E   | 3C | 00963 |        | MOVZWL                    | 2(SP)+, TYPEID_INDEX                | 8043 |
|          |    | 06 | A740 | 7F | 00966 |        | PUSHAQ                    | 6(ROUT_TBL)[R0]                     |      |
| 56       |    |    | 9E   | 9A | 0096A |        | MOVZBL                    | 2(SP)+, RESULT_TYPE                 | 8044 |
| 09       |    | 17 | AA   | 91 | 0096D |        | CMPB                      | 23(NEW_LEFT_ARG), #9                |      |
|          |    |    | 0A   | 13 | 00971 |        | BEQL                      | 114\$                               | 8045 |
| 0B       |    | 0C | AE   | E9 | 00973 |        | BLBC                      | BINARY_FLAG, 115\$                  |      |
| 09       |    | 17 | A2   | 91 | 00977 |        | CMPB                      | 23(NEW_RIGHT_ARG), #9               |      |
|          |    |    | 05   | 12 | 0097B |        | BNEQ                      | 115\$                               | 8047 |
| 59       |    |    | 01   | DD | 0097D | 114\$: | MOVL                      | #1, SD_CLASS_FLAG                   |      |
|          |    |    | 02   | 11 | 00980 |        | BRB                       | 116\$                               | 8049 |
|          |    |    | 59   | D4 | 00982 | 115\$: | CLRL                      | SD_CLASS_FLAG                       | 8051 |
| 00000052 | 8F |    | 55   | D4 | 00984 | 116\$: | CLRL                      | LENGTH                              | 8052 |
|          |    |    | 53   | D1 | 00986 |        | CMPL                      | ROUT_INDEX, #82                     |      |
| 000000FE | 8F |    | 09   | 13 | 0098D |        | BEQL                      | 117\$                               | 8053 |
|          |    |    | 53   | D1 | 0098F |        | CMPL                      | ROUT_INDEX, #254                    |      |
|          |    |    | 0B   | 12 | 00996 |        | BNEQ                      | 118\$                               | 8056 |
|          |    | 14 | AA   | 3C | 00998 | 117\$: | MOVZWL                    | 20(NEW_LEFT_ARG), LENGTH            |      |
|          |    | 14 | A2   | 3C | 0099C |        | MOVZWL                    | 20(NEW_RIGHT_ARG), R0               |      |
|          |    |    | 50   | C0 | 009A0 |        | ADDL2                     | R0, LENGTH                          |      |
|          |    |    | 55   | D5 | 009A3 | 118\$: | TSTL                      | LENGTH                              | 8058 |
|          |    |    | 33   | 12 | 009A5 |        | BNEQ                      | 121\$                               | 8061 |
| 01       |    |    | 56   | D1 | 009A7 |        | CMPL                      | RESULT_TYPE, #1                     |      |
|          |    |    | 0F   | 13 | 009AA |        | BEQL                      | 119\$                               | 8062 |
| 22       |    |    | 56   | D1 | 009AC |        | CMPL                      | RESULT_TYPE, #34                    |      |
|          |    |    | 0A   | 13 | 009AF |        | BEQL                      | 119\$                               | 8063 |
| 29       |    |    | 56   | D1 | 009B1 |        | CMPL                      | RESULT_TYPE, #41                    |      |
|          |    |    | 05   | 13 | 009B4 |        | BEQL                      | 119\$                               | 8064 |
| 2A       |    |    | 56   | D1 | 009B6 |        | CMPL                      | RESULT_TYPE, #42                    |      |
|          |    |    | 15   | 12 | 009B9 |        | BNEQ                      | 120\$                               | 8067 |
|          |    | 14 | AA   | 3C | 009BB | 119\$: | MOVZWL                    | 20(NEW_LEFT_ARG), LENGTH            |      |
|          |    | 0C | AE   | E9 | 009BF |        | BLBC                      | BINARY_FLAG, 121\$                  | 8068 |
|          |    | 14 | A2   | B1 | 009C3 |        | CMPL                      | 20(NEW_RIGHT_ARG), 20(NEW_LEFT_ARG) | 8072 |
| 14       | AA |    | 10   | 1B | 009C8 |        | BLEQU                     | 121\$                               |      |
|          |    |    | A2   | 3C | 009CA |        | MOVZWL                    | 20(NEW_RIGHT_ARG), LENGTH           | 8074 |
|          |    |    | 0A   | 11 | 009CE |        | BRB                       | 121\$                               | 8061 |
|          |    |    | 56   | DD | 009D0 | 120\$: | PUSHL                     | RESULT_TYPE                         | 8080 |
| 0000V    | CF |    | 01   | FB | 009D2 |        | CALLS                     | #1, DBG\$NUM_BYTES                  |      |
| 55       |    |    | 50   | DD | 009D7 |        | MOVL                      | R0, LENGTH                          |      |
|          |    |    | 59   | DD | 009DA | 121\$: | PUSHL                     | SD_CLASS_FLAG                       | 8087 |
|          |    |    | 7E   | D4 | 009DC |        | CLRL                      | -(SP)                               | 8084 |

[illegible]

DBGVALOP  
V04-000

8 13  
16-Sep-1984 00:32:25 VAX-11 Bliss-32 V4.0-742  
5-Sep-1984 21:54:24 [DEBUG.SRC]DBGVALOP.B32;1

Page 232  
(31)

|           |    |    |    |       |       |                               |
|-----------|----|----|----|-------|-------|-------------------------------|
| 0000V     | CF | 04 | FB | 00A68 | CALLS | #4, DBG\$PERFORM_TYPEID_CHECK |
|           | 11 | 50 | E8 | 00A6D | BLBS  | R0, 127\$                     |
|           |    | 58 | DD | 00A70 | PUSHL | R11                           |
|           |    | 01 | DD | 00A72 | PUSHL | #1                            |
|           |    | 8F | DD | 00A74 | PUSHL | #166346                       |
| 00000000G | 00 | 03 | FB | 00A7A | CALLS | #3, LIB\$SIGNAL               |
|           | 50 | AE | D0 | 00A81 | MOVL  | RESULT, R0                    |
|           |    |    | 04 | 00A85 | RET   |                               |

8130  
8137  
8138

; Routine Size: 2694 bytes, Routine Base: DBG\$CODE + 0B81

```

8037 8139 1 GLOBAL ROUTINE DBGSGET_SET_TYPEID(TYPEID, PARENT_TYPEID) =
8038 8140 1
8039 8141 1 FUNCTION
8040 8142 1 This routine is a recursive routine to get the parent type for SET data
8041 8143 1 type.
8042 8144 1
8043 8145 1 INPUTS
8044 8146 1 PARENT_TYPEID - Parent typeid. (only used for the subrange type).
8045 8147 1 TYPEID - Typeid for a given set data.
8046 8148 1
8047 8149 1 OUTPUTS
8048 8150 1 Pointer to parent typeid of set data.
8049 8151 1
8050 8152 1
8051 8153 2 BEGIN
8052 8154 2
8053 8155 2 MAP
8054 8156 2 PARENT_TYPEID: REF VECTOR[1],
8055 8157 2 TYPEID: REF RST$ENTRY;
8056 8158 2
8057 8159 2 LOCAL
8058 8160 2 PARENT_TYPE,
8059 8161 2 HIGHPTR,
8060 8162 2 LOWPTR,
8061 8163 2 SIZE;
8062 8164 2
8063 8165 2
8064 8166 2 CASE .TYPEID[RST$B_FCODE] FROM RST$K_TYPE_MINIMUM
8065 8167 2 TO RST$K_TYPE_MAXIMUM OF
8066 8168 2
8067 8169 2 SET
8068 8170 2 [RST$K_TYPE_SET]:
8069 8171 2 BEGIN
8070 8172 2 DBGSSTA_TYP_SET(.TYPEID, PARENT_TYPE, SIZE);
8071 8173 2 TYPEID = DBGSGET_SET_TYPEID(.PARENT_TYPE, PARENT_TYPEID[0]);
8072 8174 2 RETURN .TYPEID;
8073 8175 2 END;
8074 8176 2 [RST$K_TYPE_ATOMIC, RST$K_TYPE_DESCR, RST$K_TYPE_ENUM]:
8075 8177 2 RETURN .TYPEID;
8076 8178 2
8077 8179 2 [RST$K_TYPE_SUBRNG]:
8078 8180 2 BEGIN
8079 8181 2 PARENT_TYPEID[0] = .TYPEID;
8080 8182 2 DBGSSTA_TYP_SUBRNG(.TYPEID, PARENT_TYPE, LOWPTR, HIGHPTR, SIZE);
8081 8183 2 TYPEID = DBGSGET_SET_TYPEID(.PARENT_TYPE, PARENT_TYPEID[0]);
8082 8184 2 RETURN .TYPEID;
8083 8185 2 END;
8084 8186 2
8085 8187 2 [INRANGE]:
8086 8188 2 $DBG_ERROR('DBGVALOP\DBGSGET_SET_TYPEID, fcode cannot be set type');
8087 8189 2 YES;
8088 8190 2
8089 8191 2 RETURN 0;
8090 8192 1 END;

```

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|-------|
| 24 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 36    | 05E99 |
| 2C | 44 | 49 | 45 | 50 | 59 | 54 | 5F | 54 | 45 | 53 | 5F | 54 | 45 | 47    | 05EA8 |
|    |    |    |    |    |    |    |    |    |    |    | 6F | 63 | 66 | 20    | 05EB7 |
| 65 | 73 | 20 | 65 | 62 | 20 | 74 | 6F | 6E | 6E | 61 | 63 | 20 | 65 | 64    | 05EBB |
|    |    |    |    |    |    |    |    | 65 | 70 | 79 | 74 | 20 | 74 | 05ECA |       |

P.AMC: .ASCII \6DBGEVALOP\&lt;92&gt;\DBG\$GET\_SET\_TYPEID, fco\

.ASCII \de cannot be set type\

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

.ENTRY DBG\$GET\_SET\_TYPEID, Save R2

SUBL2 #16, SP  
MOVL TYPEID, R2  
CASEB 24(R2), #1, #21  
.WORD

1\$:

PUSHL SP  
PUSHAB PARENT\_TYPE  
PUSHL R2  
CALLS #3, DBG\$STA\_TYP\_SET  
BRB 5\$  
MOVL R2, R0  
RET  
MOVL R2, @PARENT\_TYPEID  
PUSHL SP  
PUSHAB HIGHPTR  
PUSHAB LOWPTR  
PUSHAB PARENT\_TYPE  
PUSHL R2  
CALLS #5, DBG\$STA\_TYP\_SUBRNG  
PUSHL PARENT\_TYPEID  
PUSHL PARENT\_TYPE  
CALLS #2, DBG\$GET\_SET\_TYPEID  
MOVL R0, TYPEID  
RET

8139

8166

8171

8172

8177

8181

8182

8183

8184

```

E 13
16-Sep-1984 00:32:25      VAX-11 B11ss-32 V4.0-742
5-Sep-1984 21:54:24      [DEBUG.SRC]DBG EVALOP.B32;1

```

Page 235  
(32)

|           |           |    |    |       |      |        |                 |
|-----------|-----------|----|----|-------|------|--------|-----------------|
|           | 00000000' | EF | 9F | 00075 | 6\$: | PUSHAB | P.AMC           |
|           |           | 01 | DD | 0007B |      | PUSHL  | #1              |
|           | 00028362  | 8F | DD | 0007D |      | PUSHL  | #164706         |
| 00000000G | 00        | 03 | FB | 00083 |      | CALLS  | #3, LIB\$SIGNAL |
|           |           | 5D | D4 | 0008A |      | CLRL   | RO              |
|           |           |    | 04 | 0008C |      | RET    |                 |

; Routine Size: 141 bytes, Routine Base: DBG\$CODE + 1607

```
8092 8193 1 GLOBAL ROUTINE DBG$GET_DTYPE (VALDESC) =
8093 8194 1
8094 8195 1 FUNCTION
8095 8196 1     Given a Value Descriptor, this routine returns a 1-byte type code
8096 8197 1     for the descriptor. This type code is ordinarily just taken from
8097 8198 1     the dtype field, but if the fcode is not atomic or descriptor, the
8098 8199 1     1-byte type code is DBG$K_MAXIMUM_DTYPE + FCODE.
8099 8200 1
8100 8201 1 INPUT
8101 8202 1     VALDESC -      A pointer to the value descriptor for which a
8102 8203 1                    type code is desired.
8103 8204 1
8104 8205 1 OUTPUT
8105 8206 1     A type code is returned.
8106 8207 1
8107 8208 1 BEGIN
8108 8209 1 MAP
8109 8210 1     VALDESC: REF DBG$VALDESC;
8110 8211 1
8111 8212 1
8112 8213 1     ! Check for data. Note that record components A.B come back with
8113 8214 1     ! KIND = TYPCOMP, and these are also data.
8114 8215 1
8115 8216 1     IF .VALDESC[DBG$B_DHDR_KIND] EQL RST$K_DATA
8116 8217 1     OR .VALDESC[DBG$B_DHDR_KIND] EQL RST$K_TYPCOMP
8117 8218 1     THEN
8118 8219 1         BEGIN
8119 8220 1
8120 8221 1             ! Cobol and Pl1 picture are treated differently, in COBOL, picture
8121 8222 1             ! is treated as 1, in PL1, we want to have FCODE + 200 as the dtype.
8122 8223 1
8123 8224 1             IF .VALDESC [DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT
8124 8225 1             THEN
8125 8226 1                 BEGIN
8126 8227 1                     IF .DBG$GB_LANGUAGE EQL DBG$K_COBOL
8127 8228 1                     THEN
8128 8229 1                         RETURN .VALDESC [DBG$B_VALUE_DTYPE]
8129 8230 1                     ELSE
8130 8231 1                         RETURN .VALDESC [DBG$B_DHDR_FCODE] + DBG$K_MAXIMUM_DTYPE;
8131 8232 1                     END;
8132 8233 1
8133 8234 1
8134 8235 1             ! The argument is data. Check the fcode.
8135 8236 1
8136 8237 1             IF .VALDESC [DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC
8137 8238 1             OR .VALDESC [DBG$B_DHDR_FCODE] EQL RST$K_TYPE_DESCR
8138 8239 1             THEN
8139 8240 1
8140 8241 1                 ! Special case for ADA fixed point and PL/1 fixed binary.
8141 8242 1                 ! These come in as
8142 8243 1                 ! class SD with the BINSKALE bit set. Return a special
8143 8244 1                 ! dtype for "FIXED" in this case.
8144 8245 1
8145 8246 1                 IF (.VALDESC[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD) AND
8146 8247 1                 (.VALDESC[DBG$B_VALUE_FL_BINSKALE])
8147 8248 1                 THEN
8148 8249 1                     RETURN DSC$K_DTYPE_FIXED
```

```
8149      8250      1
8150      8251      1
8151      8252      1
8152      8253      1
8153      8254      1
8154      8255      1
8155      8256      1
8156      8257      1
8157      8258      1
8158      8259      1
8159      8260      1
8160      8261      1
8161      8262      1
8162      8263      1
8163      8264      1
8164      8265      1
8165      8266      1
8166      8267      1
8167      8268      1
8168      8269      1
8169      8270      1
8170      8271      1
8171      8272      1
8172      8273      1
8173      8274      1
8174      8275      1
8175      8276      1
8176      8277      1
8177      8278      1
8178      8279      1
8179      8280      1
8180      8281      1
8181      8282      1
8182      8283      1
8183      8284      1
8184      8285      1
8185      8286      1
8186      8287      1
8187      8288      1
8188      8289      1
8189      8290      1

      ELSE
      ! VAX standard type.
      RETURN .VALDESC [DBG$B_VALUE_DTYPE]

      ELSE
      ! Not VAX standard. Represent type as 200 + fcode.
      RETURN .VALDESC [DBG$B_DHDR_FCODE] + DBG$K_MAXIMUM_DTYPE

      END

      ELSE
      ! Other kinds include RST$K_ROUTINE, RST$K_LINE, etc.
      ! Some languages (BLISS, MACRO) allow operations on code addresses,
      ! e.g., EVAL SUBR + 100 where SUBR is a subroutine.
      ! In this case, we expect dtype to be ZI or ZEM.
      IF .VALDESC [DBG$B_DHDR_KIND] EQL RST$K_ROUTINE
      OR .VALDESC [DBG$B_DHDR_KIND] EQL RST$K_BLOCK
      OR .VALDESC [DBG$B_DHDR_KIND] EQL RST$K_LABEL
      OR .VALDESC [DBG$B_DHDR_KIND] EQL RST$K_LINE
      OR .VALDESC [DBG$B_DHDR_KIND] EQL RST$K_ENTRY
      THEN
      ! The type should be ZI or ZEM unless an override was present.
      RETURN .VALDESC [DBG$B_VALUE_DTYPE]

      ELSE
      ! Any other kind is an error.
      BEGIN
      $DBG_ERROR ('DBGEVALOP\DBG$GET_TYPECODE unknown rst kind');
      RETURN 0;
      END;

      END;
```

.PSECT DBG\$PLIT, NOWRT, SHR, PIC, 0

.ASCII \+DBGEVALOP\<92>\DBG\$GET\_TYPECODE unknow\

.ASCII \n rst kind\

.PSECT DBG\$CODE, NOWRT, SHR, PIC, 0

.ENTRY DBG\$GET\_DTYPE, Save R2, R3  
MOVL VALDESC, R0

```
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 2B 05ED0 P.AMD:
6E 75 20 45 44 4F 43 45 50 59 54 5F 54 45 47 05EDF
                                77 6F 6E 6B 05EEE
                                64 6E 69 6B 20 74 73 72 20 6E 05EF2
```

```
50      04      000C 0000
          AC      D0 00002
```

: 8193  
: 8216

|           |           |    |    |       |        |                      |      |
|-----------|-----------|----|----|-------|--------|----------------------|------|
| 51        | 07        | A0 | 9A | 00006 | MOVZBL | 7(R0), R1            |      |
| 06        |           | 51 | 91 | 0000A | CMPB   | R1, #6               |      |
|           |           | 05 | 13 | 0000D | BEQL   | 1\$                  |      |
| 0A        |           | 51 | 91 | 0000F | CMPB   | R1, #10              | 8217 |
|           |           | 43 | 12 | 00012 | BNEQ   | 7\$                  |      |
| 52        | 06        | A0 | 9A | 00014 | MOVZBL | 6(R0), R2            | 8224 |
| 05        |           | 52 | 91 | 00018 | CMPB   | R2, #5               |      |
|           |           | 1A | 12 | 0001B | BNEQ   | 4\$                  |      |
| 03        | 00000000G | 00 | 91 | 0001D | CMPB   | DBG\$GB_LANGUAGE, #3 | 8227 |
|           |           | 06 | 12 | 00024 | BNEQ   | 2\$                  |      |
| 51        | 16        | A0 | 9A | 00026 | MOVZBL | 22(R0), R1           | 8231 |
|           |           | 07 | 11 | 0002A | BRB    | 3\$                  |      |
| 53        | 2B        | A2 | 9E | 0002C | MOVAB  | 43(R2), R3           |      |
| 51        |           | 53 | D0 | 00030 | MOVL   | R3, R1               |      |
| 50        |           | 51 | D0 | 00033 | MOVL   | R1, R0               |      |
|           |           |    | 04 | 00036 | RET    |                      |      |
| 02        |           | 52 | 91 | 00037 | CMPB   | R2, #2               | 8237 |
|           |           | 05 | 13 | 0003A | BEQL   | 5\$                  |      |
| 03        |           | 52 | 91 | 0003C | CMPB   | R2, #3               | 8238 |
|           |           | 0F | 12 | 0003F | BNEQ   | 6\$                  |      |
| 09        | 17        | A0 | 91 | 00041 | CMPB   | 23(R0), #9           | 8246 |
|           |           | 29 | 12 | 00045 | BNEQ   | 8\$                  |      |
| 24        | 1E        | A0 | E1 | 00047 | BBC    | #3, 30(R0), 8\$      | 8247 |
|           |           | 50 | 2B | 0004C | MOVL   | #43, R0              | 8249 |
|           |           |    | 04 | 0004F | RET    |                      |      |
| 52        |           | 2B | C0 | 00050 | ADDL2  | #43, R2              | 8261 |
| 50        |           | 52 | D0 | 00053 | MOVL   | R2, R0               |      |
|           |           |    | 04 | 00056 | RET    |                      | 8271 |
| 02        |           | 51 | 91 | 00057 | CMPB   | R1, #2               |      |
|           |           | 14 | 13 | 0005A | BEQL   | 8\$                  |      |
| 03        |           | 51 | 91 | 0005C | CMPB   | R1, #3               | 8272 |
|           |           | 0F | 13 | 0005F | BEQL   | 8\$                  |      |
| 04        |           | 51 | 91 | 00061 | CMPB   | R1, #4               | 8273 |
|           |           | 0A | 13 | 00064 | BEQL   | 8\$                  |      |
| 05        |           | 51 | 91 | 00066 | CMPB   | R1, #5               | 8274 |
|           |           | 05 | 13 | 00069 | BEQL   | 8\$                  |      |
| 08        |           | 51 | 91 | 0006B | CMPB   | R1, #8               | 8275 |
|           |           | 05 | 12 | 0006E | BNEQ   | 9\$                  |      |
| 50        | 16        | A0 | 9A | 00070 | MOVZBL | 22(R0), R0           | 8280 |
|           |           |    | 04 | 00074 | RET    |                      |      |
|           | 00000000' | EF | 9F | 00075 | PUSHAB | P.AMD                | 8287 |
|           |           | 01 | DD | 0007B | PUSHL  | #1                   |      |
|           | 00028362  | 8F | DD | 0007D | PUSHL  | #164706              |      |
| 00000000G | 00        | 03 | FB | 00083 | CALLS  | #3, LIB\$SIGNAL      |      |
|           |           | 50 | D4 | 0008A | CLRL   | R0                   | 8288 |
|           |           |    | 04 | 0008C | RET    |                      | 8290 |

; Routine Size: 141 bytes. Routine Base: DBG\$CODE + 1694

```

8191 8291 1 ROUTINE DBG$LANGUAGE_TYPE_CONV (CVT_ROUT_INDEX, VALUE1, VALUE2) =
8192 8292 1
8193 8293 1 FUNCTION
8194 8294 1     Performs language-specific type conversion on the given
8195 8295 1     descriptors, according to the routine index given
8196 8296 1     by CVT_ROUT_INDEX.
8197 8297 1
8198 8298 1 INPUTS
8199 8299 1     CVT_ROUT_INDEX - A routine index indicating which operation
8200 8300 1                     is to be performed. The possible values for
8201 8301 1                     this index are given in DBGLIB under CVT$K_
8202 8302 1     VALUE1          - DEBUG value descriptor for the source
8203 8303 1     VALUE2          - DEBUG value descriptor for the target
8204 8304 1
8205 8305 1 OUTPUTS
8206 8306 1     A pointer to the result descriptor is returned.
8207 8307 1
8208 8308 2 BEGIN
8209 8309 2
8210 8310 2 MAP
8211 8311 2     VALUE1 : REF DBG$VALDESC,      ! Pointer to the source value descriptor
8212 8312 2     VALUE2 : REF DBG$VALDESC;      ! Pointer to the target value descriptor
8213 8313 2
8214 8314 2 ! Case on the routine index.
8215 8315 2 !
8216 8316 2 !
8217 8317 2 CASE .CVT_ROUT_INDEX FROM CVT$K_MIN_ROUT TO CVT$K_MAX_ROUT OF
8218 8318 2 SET
8219 8319 2
8220 8320 2 [CVT$K_PL1_CVT]:
8221 8321 2     VALUE2 = PL1_TYPE_CONV(.VALUE1, .VALUE2);
8222 8322 2
8223 8323 2 [CVT$K_COB_PICT]:
8224 8324 2 BEGIN
8225 8325 2     LOCAL
8226 8326 2         BUFFER: VECTOR[4, LONG], ! Buffer area
8227 8327 2         CHANGE_SCALE: BYTE,      ! Change of scale
8228 8328 2         LANGCODE,                ! Language code
8229 8329 2         PICTPTR: REF VECTOR[BYTE],
8230 8330 2         PICTVAL,                 ! Pointer to picture representation
8231 8331 2         PSCALE: VECTOR[2, BYTE]; ! Pointer to language specific encoding
8232 8332 2         Digits and Scale
8233 8333 2
8234 8334 2
8235 8335 2 IF .VALUE2[DBG$B_DHDR_FCODE] NEQ RST$K_TYPE_PICT
8236 8336 2 THEN
8237 8337 2     $DBG_ERROR('DBGEVALOP\DBG$LANGUAGE_TYPE_CONV, fcode neq pict');
8238 8338 2
8239 8339 2 DBG$STA_TYP_PICT(.VALUE2[DBG$L_DHDR_TYPEID], LANGCODE, PICTPTR,
8240 8340 2                 PICTVAL, PSCALE);
8241 8341 2 CASE .LANGCODE FROM DBG$K_MIN_LANGUAGE TO DBG$K_MAX_LANGUAGE OF
8242 8342 2 SET
8243 8343 2     [DBG$K_COBOL]:
8244 8344 2         BEGIN
8245 8345 2             CHANGE_SCALE = .VALUE1[DBG$B_VALUE_SCALE] - .PSCALE[0];
8246 8346 2             ASHP(CHANGE_SCALE, VALUE1[DBG$B_VALUE_LENGTH],
8247 8347 2                 .VALUE1[DBG$L_VALUE_POINTER], %REF(0), PSCALE[1],
```

```

: 8248      8348      4
: 8249      8349      4
: 8250      8350      4
: 8251      8351      4
: 8252      8352      4
: 8253      8353      4
: 8254      8354      4
: 8255      8355      4
: 8256      8356      4
: 8257      8357      4
: 8258      8358      4
: 8259      8359      4
: 8260      8360      4
: 8261      8361      4
: 8262      8362      4
: 8263      8363      4
: 8264      8364      4
: 8265      8365      1

```

```

        BUFFER);
        EDITPC(PSCALE[1], BUFFER, .PICTVAL, .VALUE2[DBG$LANGUAGE_TYPE_CONV]);
        END;

        [INRANGE, OUTRANGE]:
        SIGNAL(DBG$UNIMPLENT);

        TES;

        END;

        [INRANGE, OUTRANGE]:
        $DBG_ERROR('DBGEVALOP\DBG$LANGUAGE_TYPE_CONV unknown routine index');

        TES;

        RETURN .VALUE2;
        END;

```

.PSECT DBG\$PLIT, NOWRT, SHR, PIC, 0

```

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 30 05EFC P.AME: .ASCII \0DBGEVALOP\<92>\DBG$LANGUAGE_TYPE_CONV,\
43 5F 45 50 59 54 5F 45 47 41 55 47 4E 41 4C 05F0B
                                2C 56 4E 4F 05F1A
74 63 69 70 20 71 65 6E 20 65 64 6F 63 66 20 05F1E
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 36 05F2D P.AMF: .ASCII \ fcode neg pict\
43 5F 45 50 59 54 5F 45 47 41 55 47 4E 41 4C 05F3C \6DBGEVALOP\<92>\DBG$LANGUAGE_TYPE_CONV \
                                20 56 4E 4F 05F4B
65 6E 69 74 75 6F 72 20 6E 77 6F 78 65 64 6E 69 20 05F5E .ASCII \unknown routine index\
                                78 65 64 6E 69 20 05F5E

```

.PSECT DBG\$CODE, NOWRT, SHR, PIC, 0

```

                                007C 00000 DBG$LANGUAGE TYPE CONV:
                                .WORD Save R2,R3,R4,R5,R6
                                56 00000000G 00 9E 00002 MOVAB LIB$SIGNAL, R6
                                5E 20 C2 00009 SUBL2 #32, SP
01 01 04 AC CF 0000C CASEL CMT ROUT_INDEX, #1, #1
0026 0017 00011 1$: .WORD 2$-1$, -
                                3$-1$
                                00000000' EF 9F 00015 PUSHAB P.AMF
                                01 DD 0001B PUSHL #1
                                00028362 8F DD 0001D PUSHL #164706
66 03 FB 00023 CALLS #3, LIB$SIGNAL
                                64 11 00026 BRB 7$
                                7E 08 AC 7D 00028 2$: MOVQ VALUE1, -(SP)
0000V CF 02 FB 0002C CALLS #2, PLI TYPE_CONV
OC AC 50 D0 00031 MOVL R0, VALUE2
                                75 11 00035 BRB 9$
54 0C AC D0 00037 3$: MOVL VALUE2, R4
05 06 A4 91 0003B CMPB 6(R4), #5
                                11 13 0003F BEQL 4$
                                00000000' EF 9F 00041 PUSHAB P.AME

```

```

: 8291
: 8317
: 8360
: 8321
: 8335
: 8337

```

```
K 13
16-Sep-1984 00:32:25 YAX-11 BLISS-32 V4.0-742
5-Sep-1984 21:54:24 [DEBUG.SRC]DBG EVALOP.B32:1
```

8339

8341

8353

8345

8347

8349

8364  
8365

; Routine Size: 177 bytes, Routine Base: DBG\$CODE + 1721

```
8267 8366 1 GLOBAL ROUTINE DBG$MAP_DTYPE_CLASS(IN_TYPE, FLAG) =
8268 8367 1
8269 8368 1 FUNCTION
8270 8369 1     This routine is used to obtain the class for the given dtype.
8271 8370 1     (When DEBUG override type qualifier is given, the class field
8272 8371 1     in the VMS value descriptor is zero, in order to call LIB$CVT_DX_DX,
8273 8372 1     class field must be supplied.)
8274 8373 1
8275 8374 1 INPUT
8276 8375 1     IN_TYPE      - a dtype code
8277 8376 1     FLAG         - if the flag is set that means we take SD as class
8278 8377 1                   instead of S.
8279 8378 1
8280 8379 1 OUTPUT
8281 8380 1     CLASS is returned.
8282 8381 1
8283 8382 1 BEGIN
8284 8383 1 RETURN
8285 8384 1     (CASE .IN_TYPE FROM 0 TO DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_MAXIMUM OF
8286 8385 1     SET
8287 8386 1     [DSC$K_DTYPE_Z]:
8288 8387 1         0;
8289 8388 1     [DSC$K_DTYPE_V,DSC$K_DTYPE_BU,DSC$K_DTYPE_WU,DSC$K_DTYPE_LU,
8290 8389 1     DSC$K_DTYPE_QU,DSC$K_DTYPE_B,DSC$K_DTYPE_W,DSC$K_DTYPE_C,
8291 8390 1     DSC$K_DTYPE_Q,DSC$K_DTYPE_F,DSC$K_DTYPE_D,DSC$K_DTYPE_FC,
8292 8391 1     DSC$K_DTYPE_DC,DSC$K_DTYPE_I,DSC$K_DTYPE_NU,DSC$K_DTYPE_NL,
8293 8392 1     DSC$K_DTYPE_NLO,DSC$K_DTYPE_NR,DSC$K_DTYPE_NRO,DSC$K_DTYPE_NZ,
8294 8393 1     DSC$K_DTYPE_ZI,DSC$K_DTYPE_ZEM,DSC$K_DTYPE_DSC,DSC$K_DTYPE_AD,
8295 8394 1     DSC$K_DTYPE_OU,DSC$K_DTYPE_O,DSC$K_DTYPE_G,DSC$K_DTYPE_H,
8296 8395 1     DSC$K_DTYPE_GC,DSC$K_DTYPE_HC,DSC$K_DTYPE_CIT,DSC$K_DTYPE_BPV,
8297 8396 1     DSC$K_DTYPE_BLV,DSC$K_DTYPE_SV]:
8298 8397 1     IF FLAG THEN DSC$K_CLASS_SD ELSE DSC$K_CLASS_S;
8299 8398 1     [DSC$K_DTYPE_VU,DSC$K_DTYPE_TF,DSC$K_DTYPE_SVU]:
8300 8399 1         DSC$K_CLASS_UBS;
8301 8400 1     [DSC$K_DTYPE_VT,DSC$K_DTYPE_AC,DSC$K_DTYPE_AZ]:
8302 8401 1         DSC$K_CLASS_VS;
8303 8402 1     [DSC$K_DTYPE_P]:
8304 8403 1         DSC$K_CLASS_SD;
8305 8404 1     [DSC$K_DTYPE_FIXED]:
8306 8405 1         DSC$K_CLASS_SD;
8307 8406 1     [INRANGE]:
8308 8407 1         DSC$K_CLASS_Z;
8309 8408 1     [OUTRANGE]:
8310 8409 1         $DBG_ERROR ('DBG$EVALOP\DBG$MAP_DTYPE_CLASS');
8311 8410 1     TES);
8312 8411 1 END;
```

.PSECT DBG\$PLIT,NOWRT, SHR, PIC,0

```
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 1D 05F64 P.AMG: .ASCII <29>\DBG$EVALOP\<92>\DBG$MAP_DTYPE_CLAS\
53 41 4C 43 5F 45 50 59 54 44 5F 50 41 4D 05F73
53 05F81 .ASCII \S\
```



8  
3  
2  
6  
5  
4  
8  
7

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBG EVALOP.B32;1

Page 244  
(35)

| Address  | Hex | Assembly | Comment         |
|----------|-----|----------|-----------------|
| 00000000 | 00  | PUSHAB   | P.AMG           |
| 00000000 | 00  | PUSHL    | #1              |
| 00000000 | 00  | PUSHL    | #164706         |
| 00000000 | 00  | CALLS    | #3, LIB\$SIGNAL |
| 00000000 | 00  | RET      |                 |
| 00000000 | 00  | BLBS     | FLAG, 5\$       |
| 00000000 | 00  | MOVL     | #1, R0          |
| 00000000 | 00  | RET      |                 |
| 00000000 | 00  | MOVL     | #13, R0         |
| 00000000 | 00  | RET      |                 |
| 00000000 | 00  | MOVL     | #11, R0         |
| 00000000 | 00  | RET      |                 |
| 00000000 | 00  | MOVL     | #9, R0          |
| 00000000 | 00  | RET      |                 |
| 00000000 | 00  | CLRL     | R0              |
| 00000000 | 00  | RET      |                 |

; Routine Size: 188 bytes, Routine Base: DBG\$CODE + 17D2

```
8314 8412 1 GLOBAL ROUTINE DBG$NUM_BYTES(IN_TYPE) =
8315 8413 1
8316 8414 1 FUNCTION
8317 8415 1     This routine is used to obtain the number of bytes of storage that
8318 8416 1     are needed to hold data of a given dtype.
8319 8417 1
8320 8418 1 INPUT
8321 8419 1     IN_TYPE          - a dtype code
8322 8420 1
8323 8421 1 OUTPUT
8324 8422 1     The number of bytes of storage needed to hold data of type dtype
8325 8423 1     is returned.
8326 8424 1
8327 8425 2 BEGIN
8328 8426 2 RETURN
8329 8427 2     (CASE .IN_TYPE FROM 0 TO DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_MAXIMUM OF
8330 8428 2     SET
8331 8429 2         [DSC$K_DTYPE_Z]           : 4;
8332 8430 2         [DSC$K_DTYPE_BU]          : 1;
8333 8431 2         [DSC$K_DTYPE_B]           : 1;
8334 8432 2         [DSC$K_DTYPE_WU]          : 2;
8335 8433 2         [DSC$K_DTYPE_W]           : 2;
8336 8434 2         [DSC$K_DTYPE_LU]          : 4;
8337 8435 2         [DSC$K_DTYPE_L]           : 4;
8338 8436 2         [DSC$K_DTYPE_QU]          : 4;
8339 8437 2         [DSC$K_DTYPE_Q]           : 8;
8340 8438 2         [DSC$K_DTYPE_OU]          : 16;
8341 8439 2         [DSC$K_DTYPE_O]           : 16;
8342 8440 2         [DSC$K_DTYPE_F]           : 4;
8343 8441 2         [DSC$K_DTYPE_D]           : 8;
8344 8442 2         [DSC$K_DTYPE_G]           : 8;
8345 8443 2         [DSC$K_DTYPE_H]           : 16;
8346 8444 2         [DSC$K_DTYPE_FC]          : 8;
8347 8445 2         [DSC$K_DTYPE_DC]          : 16;
8348 8446 2         [DSC$K_DTYPE_GC]          : 16;
8349 8447 2         [DSC$K_DTYPE_HC]          : 32;
8350 8448 2         [DSC$K_DTYPE_P]           : 31;
8351 8449 2         [DSC$K_DTYPE_TF]          : 1;
8352 8450 2         [DSC$K_DTYPE_FIXED]        : 4;
8353 8451 2         [DBG$K_DTYPE_ENUM]         : 4;
8354 8452 2         [DBG$K_DTYPE_TPTR]         : 4;
8355 8453 2         [DBG$K_DTYPE_PTR]          : 4;
8356 8454 2         [INRANGE]                 : 64; ! Default to 64 bytes
8357 8455 2         [OUTRANGE]                : $DBG_ERROR ('DBGEVALOP\DBG$NUM_BYTES');
8358 8456 2     TES);
8359 8457 1 END;
```

```
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 17 05F82 P.AMH: .ASCII <23>\DBGEVALOP\<92>\DBG$NUM_BYTES\
53 45 54 59 42 5F 4D 55 4E 05F91
```

```
.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
```

```
.PSECT DBG$CODE,NOWRT, SHR, PIC,0
```

Page 246  
(36)

[illegible]

D 14  
16-Sep-1984 00:32:25 VAX-11 Bliss-32 V4.0-742  
5-Sep-1984 21:54:24 [DEBUG.SRC]DBG EVALOP.B32;1

95-15,-  
95-15,-  
95-15,-  
95-15,-  
95-15,-  
85-15,-  
95-15,-  
95-15,-  
95-15,-  
95-15,-  
95-15,-  
95-15,-

| Address  | Instruction | Comment         |
|----------|-------------|-----------------|
| 00000000 | PUSHAB      | P.AMH           |
| 00000001 | PUSHL       | #1              |
| 00000002 | PUSHL       | #164706         |
| 00000003 | CALLS       | #3, LIB\$SIGNAL |
| 00000004 | RET         |                 |
| 00000005 | MOVL        | #2, R0          |
| 00000006 | RET         |                 |
| 00000008 | MOVL        | #8, R0          |
| 00000009 | RET         |                 |
| 00000010 | MOVL        | #16, R0         |
| 00000011 | RET         |                 |
| 00000020 | MOVL        | #32, R0         |
| 00000021 | RET         |                 |
| 0000001F | MOVL        | #31, R0         |
| 00000020 | RET         |                 |
| 00000001 | MOVL        | #1, R0          |
| 00000021 | RET         |                 |
| 00000004 | MOVL        | #4, R0          |
| 00000022 | RET         |                 |
| 0000008F | MOVZBL      | #64, R0         |
| 00000090 | RET         |                 |

8427

8457

; Routine Size: 198 bytes, Routine Base: DBG\$CODE + 188E

```
8361 8458 1 GLOBAL ROUTINE DBG$PERFORM_TYPEID_CHECK(TYPE_INDEX, LEFT_ARG, RIGHT_ARG, RESULT) =
8362 8459 1
8363 8460 1 FUNCTION
8364 8461 1     This routine performs type check on the arguments according to
8365 8462 1     the given type index. This routine requires to have left argument,
8366 8463 1     and one of the right argument or result, or both. For non-atomic data
8367 8464 1     items, left argument, right argument must have TYPEID. For non-atomic
8368 8465 1     data item, if the result's typeid is zero, then TYPEID is taken from
8369 8466 1     left argument. (This only should occur after MAKE_VAL_DESC is called
8370 8467 1     to create the result value descriptor). TYPEID check will be performed
8371 8468 1     on left argument and right argument for non-atomic data types. And/Or
8372 8469 1     TYPEID check will be performed on left argument and result, and also
8373 8470 1     range check is done on the result.
8374 8471 1
8375 8472 1     This routine is called from DEPOSIT command with left_arg (src) and
8376 8473 1     right_arg (dst) to perform the typeid check before the DEPOSIT.
8377 8474 1     Then it is also called from DEPOSIT command with left_arg
8378 8475 1     (src) and result (dst) to perform the range check after the DEPOSIT.
8379 8476 1
8380 8477 1     This routine is called from EV command to have standard left_arg,
8381 8478 1     or right_arg (operands) and result.
8382 8479 1
8383 8480 1 INPUTS
8384 8481 1     TYPE_INDEX      - Type check index.
8385 8482 1
8386 8483 1     LEFT_ARG        - Left argument value descriptor. This argument
8387 8484 1     must have typeid for non-atomic data types.
8388 8485 1
8389 8486 1     RIGHT_ARG       - Right argument value descriptor. This argument
8390 8487 1     must have typeid for non-atomic data types. This
8391 8488 1     argument may not present.
8392 8489 1
8393 8490 1     RESULT          - Result argument value descriptor. This argument
8394 8491 1     may not present. Its typeid may be zero for non-
8395 8492 1     atomic data types.
8396 8493 1
8397 8494 1 OUTPUTS
8398 8495 1     Returned status:
8399 8496 1
8400 8497 1     TRUE or FALSE
8401 8498 1
8402 8499 1
8403 8500 2 BEGIN
8404 8501 2
8405 8502 2 MAP
8406 8503 2     LEFT_ARG: REF DBG$VALDESC,      ! Pointer to left argument
8407 8504 2     RIGHT_ARG: REF DBG$VALDESC,    ! Pointer to right argument
8408 8505 2     RESULT: REF DBG$VALDESC;       ! Pointer to result argument
8409 8506 2
8410 8507 2 LOCAL
8411 8508 2     DUMMY1,
8412 8509 2     DUMMY2,
8413 8510 2     LEFT_DTYPE,                ! Left arg's data type
8414 8511 2     LEFT_FCODE,               ! Left arg's fcode
8415 8512 2     LEFT_TYPEID: REF RST$ENTRY,  ! Left arg's typeid
8416 8513 2     PARENT_TYPE: REF RST$ENTRY,  ! Parent typeid for subrange
8417 8514 2     RESULT_DTYPE,             ! Result's data type
```

```

8418      8515      2      RESULT_FCODE,      ! Result's fcode
8419      8516      2      RESULT_TYPEID: REF RST$ENTRY,      ! Result's typeid
8420      8517      2      RIGHT_DTYPE,      ! Right arg's data type
8421      8518      2      RIGHT_FCODE,      ! Right arg's fcode
8422      8519      2      RIGHT_TYPEID: REF RST$ENTRY,      ! Right arg's typeid
8423      8520      2      SIZE,      ! The size of the elements
8424      8521      2      STATUS;      ! TRUE or FALSE
8425      8522      2
8426      8523      2
8427      8524      2      ! Make sure left argument is supplied.
8428      8525      2      !
8429      8526      2      IF .LEFT_ARG EQL 0
8430      8527      2      THEN
8431      8528      2          $DBG_ERROR('DBGEVALOP\DBG$PERFORM_TYPEID_CHECK');
8432      8529      2
8433      8530      2
8434      8531      2      ! Make sure left arg has TYPEID for non-atomic data types.
8435      8532      2      !
8436      8533      2      IF NOT ((.LEFT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC) OR
8437      8534      2          (.LEFT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_DESCR))
8438      8535      2      THEN
8439      8536      2          IF .LEFT_ARG[DBG$L_DHDR_TYPEID] EQL 0
8440      8537      2          THEN
8441      8538      2              $DBG_ERROR('DBGEVALOP\DBG$PERFORM_TYPEID_CHECK, no typeid for non-atomic data');
8442      8539      2
8443      8540      2
8444      8541      2      ! If there is right argument, make sure right arg has TYPEID for non-atomic
8445      8542      2      ! data types. (ie, for unary operand).
8446      8543      2      !
8447      8544      2      IF .RIGHT_ARG NEQ 0
8448      8545      2      THEN
8449      8546      2          BEGIN
8450      8547      2              IF NOT ((.RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC) OR
8451      8548      2                  (.RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_DESCR))
8452      8549      2              THEN
8453      8550      2                  IF .RIGHT_ARG[DBG$L_DHDR_TYPEID] EQL 0
8454      8551      2                  THEN
8455      8552      2                      $DBG_ERROR('DBGEVALOP\DBG$PERFORM_TYPEID_CHECK, no typeid for non-atomic data');
8456      8553      2
8457      8554      2          END;
8458      8555      2
8459      8556      2
8460      8557      2      ! One may only want to perform TYPEID check on the operands. In this
8461      8558      2      ! case, there is no need to have result operand.
8462      8559      2      !
8463      8560      2      IF .RIGHT_ARG EQL 0 AND .RESULT EQL 0
8464      8561      2      THEN
8465      8562      2          $DBG_ERROR('DBGEVALOP\DBG$PERFORM_TYPEID_CHECK');
8466      8563      2
8467      8564      2
8468      8565      2      ! Perform type check.
8469      8566      2      !
8470      8567      2      STATUS = TRUE;
8471      8568      2      PARENT_TYPE = 0;
8472      8569      2      CASE .TYPE_INDEX FROM ORT$K_TYPEID_MIN_ROUT TO ORT$K_TYPEID_MAX_ROUT OF
8473      8570      2          SET
8474      8571      2          [ORT$K_TYPEID_ENUM_ENUM];
```

```
8475 8572 3
8476 8573 3
8477 8574 3
8478 8575 4
8479 8576 4
8480 8577 4
8481 8578 4
8482 8579 4
8483 8580 4
8484 8581 4
8485 8582 4
8486 8583 4
8487 8584 4
8488 8585 4
8489 8586 5
8490 8587 5
8491 8588 5
8492 8589 5
8493 8590 5
8494 8591 4
8495 8592 5
8496 8593 5
8497 8594 5
8498 8595 6
8499 8596 7
8500 8597 7
8501 8598 6
8502 8599 6
8503 8600 6
8504 8601 6
8505 8602 5
8506 8603 4
8507 8604 3
8508 8605 3
8509 8606 3
8510 8607 3
8511 8608 3
8512 8609 3
8513 8610 3
8514 8611 3
8515 8612 3
8516 8613 3
8517 8614 3
8518 8615 3
8519 8616 3
8520 8617 3
8521 8618 3
8522 8619 3
8523 8620 3
8524 8621 3
8525 8622 3
8526 8623 3
8527 8624 4
8528 8625 4
8529 8626 4
8530 8627 4
8531 8628 4
```

```
BEGIN
IF .RIGHT_ARG NEQ 0
THEN
  BEGIN
    ! If this is called from DEPOSIT command, if RESULT is not
    ! presented, this must be one of the sanity check to make
    ! sure DEPOSIT is possible. Note: DRINK = 1 is valid
    ! case, so in here we allow this case to be legal.
    ! (LEFT_ARG is the source, RIGHT_ARG is the target).
    IF .LEFT_ARG[DBG$SL_DHDR_TYPEID] NEQ 0
    THEN
      BEGIN
        STATUS = TYPEID_CHECK_ENUM(.LEFT_ARG[DBG$SL_DHDR_TYPEID],
        .RIGHT_ARG[DBG$SL_DHDR_TYPEID]);
      END
    ELSE
      BEGIN
        IF .RESULT EQL 0
        THEN
          BEGIN
            IF (.LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_L OR
            .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_LU)
            THEN
              STATUS = TRUE
            ELSE
              STATUS = FALSE;
            END;
          END;
        END;
      END;
    END;
    ! If there is no result, or result is atomic data, or left and
    ! right TYPEID check failed, returns.
    IF (.RESULT EQL 0) OR
    (.RESULT[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC) OR
    (.RESULT[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_DESCR) OR
    NOT .STATUS
    THEN
      RETURN .STATUS;

    ! Perform result TYPEID check and range check.
    ! CASE .RESULT[DBG$B_DHDR_FCODE] FROM RST$K_TYPE_MINIMUM
    ! TO RST$K_TYPE_MAXIMUM OF
    SET
    [RST$K_TYPE_ENUM]:
    BEGIN
      IF .RESULT[DBG$SL_DHDR_TYPEID] EQL 0
      THEN
        RESULT[DBG$SL_DHDR_TYPEID] = .LEFT_ARG[DBG$SL_DHDR_TYPEID];
```

```

8532 8629 4
8533 8630 4
8534 8631 5
8535 8632 5
8536 8633 5
8537 8634 5
8538 8635 5
8539 8636 4
8540 8637 5
8541 8638 6
8542 8639 6
8543 8640 5
8544 8641 5
8545 8642 4
8546 8643 4
8547 8644 4
8548 8645 4
8549 8646 4
8550 8647 4
8551 8648 4
8552 8649 4
8553 8650 4
8554 8651 4
8555 8652 4
8556 8653 2
8557 8654 2
8558 8655 2
8559 8656 2
8560 8657 4
8561 8658 4
8562 8659 4
8563 8660 4
8564 8661 4
8565 8662 4
8566 8663 4
8567 8664 4
8568 8665 4
8569 8666 4
8570 8667 4
8571 8668 5
8572 8669 5
8573 8670 5
8574 8671 4
8575 8672 4
8576 8673 5
8577 8674 5
8578 8675 5
8579 8676 5
8580 8677 4
8581 8678 4
8582 8679 4
8583 8680 4
8584 8681 4
8585 8682 4
8586 8683 5
8587 8684 5
8588 8685 5

IF .LEFT_ARG[DBG$L_DHDR_TYPEID] NEQ 0
THEN
  BEGIN
    STATUS = TYPEID_CHECK_ENUM(.LEFT_ARG[DBG$L_DHDR_TYPEID],
                                .RESULT[DBG$L_DHDR_TYPEID]);
  END
ELSE
  BEGIN
    IF NOT (.LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_L OR
            .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_LU)
    THEN
      STATUS = FALSE;
    END;
  IF NOT .STATUS THEN RETURN .STATUS;
  RETURN TYPEID_RANGE_CHECK_ENUM(.LEFT_ARG, .RESULT[DBG$L_DHDR_TYPEID]);
END;

[INRANGE, OUTRANGE]:
  STATUS = FALSE;
TES;

END;

[ORT$K_TYPEID_SET_SET]:
  BEGIN
    LEFT_TYPEID = .LEFT_ARG[DBG$L_DHDR_TYPEID];
    LEFT_FCODE = .LEFT_ARG[DBG$B_DHDR_FCODE];
    LEFT_DTYPE = .LEFT_ARG[DBG$B_VALUE_DTYPE];
    IF .LEFT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_SET
    THEN
      BEGIN
        LEFT_TYPEID = DBG$GET_SET_TYPEID(
          .LEFT_ARG[DBG$L_DHDR_TYPEID], PARENT_TYPE);
        LEFT_FCODE = .LEFT_TYPEID[RST$B_FCODE];
        IF .LEFT_FCODE EQL RST$K_TYPE_ATOMIC
        THEN
          BEGIN
            DBG$STA_TYP_ATOMIC(.LEFT_TYPEID, LEFT_DTYPE, SIZE);
            IF .LEFT_DTYPE EQL DST$K_BOOL THEN LEFT_DTYPE = DSC$K_DTYPE_TF;
          END;
        END;
      END;
    IF .RIGHT_ARG NEQ 0
    THEN
      BEGIN
        RIGHT_TYPEID = .RIGHT_ARG[DBG$L_DHDR_TYPEID];
        RIGHT_FCODE = .RIGHT_ARG[DBG$B_DHDR_FCODE];
        RIGHT_DTYPE = .RIGHT_ARG[DBG$B_VALUE_DTYPE];
        IF .RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_SET
        THEN
          BEGIN
            RIGHT_TYPEID = DBG$GET_SET_TYPEID(
              .RIGHT_ARG[DBG$L_DHDR_TYPEID], PARENT_TYPE);

```

|      |      |   |
|------|------|---|
| 8589 | 8686 | 5 |
| 8590 | 8687 | 5 |
| 8591 | 8688 | 5 |
| 8592 | 8689 | 6 |
| 8593 | 8690 | 6 |
| 8594 | 8691 | 6 |
| 8595 | 8692 | 5 |
| 8596 | 8693 | 5 |
| 8597 | 8694 | 4 |
| 8598 | 8695 | 4 |
| 8599 | 8696 | 4 |
| 8600 | 8697 | 4 |
| 8601 | 8698 | 4 |
| 8602 | 8699 | 3 |
| 8603 | 8700 | 3 |
| 8604 | 8701 | 3 |
| 8605 | 8702 | 3 |
| 8606 | 8703 | 3 |
| 8607 | 8704 | 3 |
| 8608 | 8705 | 3 |
| 8609 | 8706 | 3 |
| 8610 | 8707 | 3 |
| 8611 | 8708 | 3 |
| 8612 | 8709 | 3 |
| 8613 | 8710 | 3 |
| 8614 | 8711 | 3 |
| 8615 | 8712 | 4 |
| 8616 | 8713 | 4 |
| 8617 | 8714 | 4 |
| 8618 | 8715 | 4 |
| 8619 | 8716 | 4 |
| 8620 | 8717 | 4 |
| 8621 | 8718 | 4 |
| 8622 | 8719 | 4 |
| 8623 | 8720 | 4 |
| 8624 | 8721 | 4 |
| 8625 | 8722 | 4 |
| 8626 | 8723 | 4 |
| 8627 | 8724 | 5 |
| 8628 | 8725 | 5 |
| 8629 | 8726 | 5 |
| 8630 | 8727 | 4 |
| 8631 | 8728 | 4 |
| 8632 | 8729 | 4 |
| 8633 | 8730 | 4 |
| 8634 | 8731 | 4 |
| 8635 | 8732 | 4 |
| 8636 | 8733 | 4 |
| 8637 | 8734 | 4 |
| 8638 | 8735 | 4 |
| 8639 | 8736 | 4 |
| 8640 | 8737 | 4 |
| 8641 | 8738 | 4 |
| 8642 | 8739 | 4 |
| 8643 | 8740 | 4 |
| 8644 | 8741 | 4 |
| 8645 | 8742 | 4 |

```
RIGHT_FCODE = .RIGHT_TYPEID[RST$B_FCODE];
IF .RIGHT_FCODE EQL RST$K_TYPE_ATOMIC
THEN
  BEGIN
    DBG$STA_TYP_ATOMIC(.RIGHT_TYPEID, RIGHT_DTYPE, SIZE);
    IF .RIGHT_DTYPE EQL DST$K_BOOL THEN RIGHT_DTYPE = DSC$K_DTYPE_TF;
  END;
END;

STATUS = TYPEID_CHECK_SET(.LEFT_TYPEID, .RIGHT_TYPEID,
                          .LEFT_FCODE, .RIGHT_FCODE,
                          .LEFT_DTYPE, .RIGHT_DTYPE);
END;

IF (.RESULT EQL 0) OR
   (.RESULT[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC) OR
   (.RESULT[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_DESCR) OR
   NOT .STATUS
THEN
  RETURN .STATUS;

CASE .RESULT[DBG$B_DHDR_FCODE] FROM RST$K_TYPE_MINIMUM
TO RST$K_TYPE_MAXIMUM OF
  SET
  [RST$K_TYPE_SET]:
    BEGIN
      RESULT_FCODE = .RESULT[DBG$B_DHDR_FCODE];
      RESULT_DTYPE = .RESULT[DBG$B_VALUE_DTYPE];
      IF .RESULT[DBG$L_DHDR_TYPEID] EQL 0
      THEN
        RESULT[DBG$L_DHDR_TYPEID] = .LEFT_ARG[DBG$L_DHDR_TYPEID];

      RESULT_TYPEID = DBG$GET_SET_TYPEID(
        .RESULT[DBG$L_DHDR_TYPEID], PARENT_TYPE);
      RESULT_FCODE = .RESULT_TYPEID[RST$B_FCODE];
      IF .RESULT_FCODE EQL RST$K_TYPE_ATOMIC
      THEN
        BEGIN
          DBG$STA_TYP_ATOMIC(.RESULT_TYPEID, RESULT_DTYPE, SIZE);
          IF .RESULT_DTYPE EQL DST$K_BOOL THEN RESULT_DTYPE = DSC$K_DTYPE_TF;
        END;

      STATUS = TYPEID_CHECK_SET(.LEFT_TYPEID, .RESULT_TYPEID,
                                .LEFT_FCODE, .RESULT_FCODE,
                                .LEFT_DTYPE, .RESULT_DTYPE);

      IF NOT .STATUS THEN RETURN .STATUS;

      ! Set the typeid for subrange. Typeid is set in routine
      ! DBG$GET_SET_TYPEID.
      IF .PARENT_TYPE NEQ 0
      THEN
        IF .PARENT_TYPE[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG
        THEN
```

```

: 8646      8743      4      RESULT_TYPEID = .PARENT_TYPE;
: 8647      8744      4
: 8648      8745      4
: 8649      8746      4      ! Perform range check.
: 8650      8747      4
: 8651      8748      4      SELECTONE .RESULT_TYPEID[RST$B_FCODE] OF
: 8652      8749      4      SET
: 8653      8750      4      [RST$K_TYPE_SUBRNG]:
: 8654      8751      4      RETURN TYPEID_RANGE_CHECK_SUBRNG(.LEFT_ARG, .RESULT_TYPEID);
: 8655      8752      4
: 8656      8753      4      TES;
: 8657      8754      4      END;
: 8658      8755      4
: 8659      8756      4      [INRANGE, OTRANGE]:
: 8660      8757      4      STATUS = FALSE;
: 8661      8758      4      TES;
: 8662      8759      4
: 8663      8760      4      END;
: 8664      8761      4
: 8665      8762      4
: 8666      8763      4      ! Typeid pointer typeid check would be always called from DEPOSIT
: 8667      8764      4      command with Left_arg (src) and Right_arg (dst). There is no
: 8668      8765      4      result. It turns out this is not supported. For we have mapped
: 8669      8766      4      the TPTR into LU, so there is no typeid check required.
: 8670      8767      4
: 8671      8768      4      [ORT$K_TYPEID_TPTR_TPTR]:
: 8672      8769      4      BEGIN
: 8673      8770      4      0;
: 8674      8771      4      END;
: 8675      8772      4
: 8676      8773      4
: 8677      8774      4      ! Subrange typeid check would be always called from DEPOSIT
: 8678      8775      4      command with Left_arg (src) and Right_arg (dst) for typeid
: 8679      8776      4      check and also called with Left_arg (src) and Result (dst) for
: 8680      8777      4      range check.
: 8681      8778      4
: 8682      8779      4      [ORT$K_TYPEID_SUBRNG_SUBRNG]:
: 8683      8780      4      BEGIN
: 8684      8781      4      IF .RESULT EQL 0
: 8685      8782      4      THEN
: 8686      8783      4      BEGIN
: 8687      8784      4      PARENT_TYPE = .RIGHT_ARG[DBG$SL_DHDR_TYPEID];
: 8688      8785      4      WHILE .PARENT_TYPE[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG DO
: 8689      8786      4      DBG$STA_TYP_SUBRNG(.PARENT_TYPE, PARENT_TYPE, DUMMY1, DUMMY2, SIZE);
: 8690      8787      4
: 8691      8788      4      SELECTONE .PARENT_TYPE[RST$B_FCODE] OF
: 8692      8789      4      SET
: 8693      8790      4      [RST$K_TYPE_ENUM]:
: 8694      8791      4      BEGIN
: 8695      8792      4      IF .LEFT_ARG[DBG$SL_DHDR_TYPEID] NEQ 0
: 8696      8793      4      THEN
: 8697      8794      4      BEGIN
: 8698      8795      4      STATUS = TYPEID_CHECK_ENUM(.LEFT_ARG[DBG$SL_DHDR_TYPEID],
: 8699      8796      4      .PARENT_TYPE);
: 8700      8797      4      END
: 8701      8798      4
: 8702      8799      4      ELSE
```

```

: 8703      8800      6
: 8704      8801      7
: 8705      8802      7
: 8706      8803      6
: 8707      8804      6
: 8708      8805      5
: 8709      8806      4
: 8710      8807      4
: 8711      8808      4
: 8712      8809      4
: 8713      8810      4
: 8714      8811      4
: 8715      8812      4
: 8716      8813      4
: 8717      8814      4
: 8718      8815      4
: 8719      8816      4
: 8720      8817      4
: 8721      8818      4
: 8722      8819      4
: 8723      8820      4
: 8724      8821      4
: 8725      8822      4
: 8726      8823      4
: 8727      8824      4
: 8728      8825      4
: 8729      8826      4
: 8730      8827      4
: 8731      8828      4
: 8732      8829      2
: 8733      8830      2
: 8734      8831      2
: 8735      8832      2
: 8736      8833      2
: 8737      8834      2
: 8738      8835      2
: 8739      8836      1

      BEGIN
      IF NOT (.LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_L OR
              .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_LU)
      THEN
          STATUS = FALSE;
      END;
      END;
      [RST$K_TYPE_ATOMIC]:
      BEGIN
      DBG$STA TYP ATOMIC(.PARENT_TYPE, RESULT DTYPE, SIZE);
      IF .RESULT_DTYPE EQL DST$K_BOOL THEN RESULT_DTYPE = DSC$K_DTYPE_TF;
      IF .RESULT_DTYPE EQL .LEFT_ARG[DBG$B_VALUE_DTYPE]
      THEN
          STATUS = TRUE
      ELSE
          STATUS = FALSE;
      END;
      [OTHERWISE]:
      STATUS = FALSE;
      TES;
      RETURN .STATUS;
      END
      ELSE
      RETURN TYPEID_RANGE_CHECK_SUBRNG(.LEFT_ARG, .RESULT[DBG$L_DHDR_TYPEID]);
      END;
      [INRANGE, OUTRANGE]:
      $DBG_ERROR('DBGEVALOP\DBG$PERFORM_TYPEID_CHECK, unknown type check index');
      TES;
      RETURN .STATUS;
      END;
```

.PSECT DBG\$PLIT, NOWRT, SHR, PIC, 0

```

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 22 05F9A P.AMI: .ASCII \DBGEVALOP\<92>\DBG$PERFORM_TYPEID_CHECK\
5F 44 49 45 50 59 54 5F 4D 52 4F 46 52 45 50 05FA9
                                     43 45 48 43 05FB8
                                     48 05FBC
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 41 05FBD P.AMJ: .ASCII \K\
5F 44 49 45 50 59 54 5F 4D 52 4F 46 52 45 50 05FCC .ASCII \ADBGEVALOP\<92>\DBG$PERFORM_TYPEID_CHECK\
                                     43 45 48 43 05FDB
                                     05FDF .ASCII \K, no typeid for non-atomic data\
6F 66 20 64 69 65 70 79 74 20 6F 6E 20 2C 48 05FEE
61 64 20 63 69 6D 6F 74 61 2D 6E 6F 6E 20 72 05FFD
                                     61 74 05FFF
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 41 05FFF P.AMK: .ASCII \ADBGEVALOP\<92>\DBG$PERFORM_TYPEID_CHECK\
5F 44 49 45 50 59 54 5F 4D 52 4F 46 52 45 50 0600E
                                     43 45 48 43 0601D
6F 66 20 64 69 65 70 79 74 20 6F 6E 20 2C 48 06021 .ASCII \K, no typeid for non-atomic data\
61 64 20 63 69 6D 6F 74 61 2D 6E 6F 6E 20 72 06030
```

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| 24 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 74 | 0603F |
| 5F | 44 | 49 | 45 | 50 | 59 | 54 | 5F | 4D | 52 | 4F | 46 | 52 | 45 | 50 | 06041 |
|    |    |    |    |    |    |    |    |    |    |    | 43 | 45 | 48 | 43 | 06050 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    | 48 | 0605F |
| 24 | 47 | 42 | 44 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 3C | 06063 |
| 5F | 44 | 49 | 45 | 50 | 59 | 54 | 5F | 4D | 52 | 4F | 46 | 52 | 45 | 50 | 06064 |
|    |    |    |    |    |    |    |    |    |    |    | 43 | 45 | 48 | 43 | 06073 |
| 65 | 70 | 79 | 74 | 20 | 6E | 77 | 6F | 6E | 6B | 6E | 75 | 20 | 2C | 4B | 06082 |
|    |    |    | 78 | 65 | 64 | 6E | 69 | 20 | 6B | 63 | 65 | 68 | 63 | 20 | 06086 |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 06095 |

P.AHL: .ASCII \DBGEVALOP\<92>\DBG\$PERFORM\_TYPEID\_CHECK\

P.AMH: .ASCII \K\<DBGEVALOP\<92>\DBG\$PERFORM\_TYPEID\_CHECK\

.ASCII \K, unknown type check index\

.PSECT DBG\$CODE, NOWRT, SHR, PIC, 0

.ENTRY DBG\$PERFORM\_TYPEID\_CHECK, Save R2,R3,R4,R5,-, 8458

|    |           |      |               |
|----|-----------|------|---------------|
|    |           | OFFC | 00000         |
| 5B | 00000000G | 00   | 9E 00002      |
| 5A | 00000000' | EF   | 9E 00009      |
| 59 | 00000000G | 00   | 9E 00010      |
| 5E |           | 1C   | C2 00017      |
| 52 | 08        | AC   | D0 0001A      |
|    |           | 0D   | 12 0001E      |
|    |           | 5A   | DD 00020      |
|    |           | 01   | DD 00022      |
|    | 00028362  | 8F   | DD 00024      |
| 69 |           | 03   | FB 0002A      |
| 02 | 06        | A2   | 91 0002D 1\$: |
|    |           | 19   | 13 00031      |
| 03 | 06        | A2   | 91 00033      |
|    |           | 13   | 13 00037      |
|    | 08        | A2   | D5 00039      |
|    |           | 0E   | 12 0003C      |
|    | 23        | AA   | 9F 0003E      |
|    |           | 01   | DD 00041      |
|    | 00028362  | 8F   | DD 00043      |
| 69 |           | 03   | FB 00049      |
| 54 | 0C        | AC   | D0 0004C 2\$: |
|    |           | 56   | D4 00050      |
|    |           | 54   | D5 00052      |
|    |           | 21   | 13 00054      |
|    |           | 56   | D6 00056      |
| 02 | 06        | A4   | 91 00058      |
|    |           | 19   | 13 0005C      |
| 03 | 06        | A4   | 91 0005E      |
|    |           | 13   | 13 00062      |
|    | 08        | A4   | D5 00064      |
|    |           | 0E   | 12 00067      |
|    | 65        | AA   | 9F 00069      |
|    |           | 01   | DD 0006C      |
|    | 00028362  | 8F   | DD 0006E      |
| 69 |           | 03   | FB 00074      |
|    |           | 54   | D5 00077 3\$: |
|    |           | 14   | 12 00079      |
|    | 10        | AC   | D5 0007B      |
|    |           | 0F   | 12 0007E      |
|    | 00A7      | CA   | 9F 00080      |

|        |                          |      |
|--------|--------------------------|------|
| MOVAB  | DBG\$STA_TYP_ATOMIC, R11 |      |
| MOVAB  | P.AMI, R10               |      |
| MOVAB  | LIB\$SIGNAL, R9          |      |
| SUBL2  | #28, SP                  |      |
| MOVL   | LEFT_ARG, R2             | 8526 |
| BNEQ   | 1\$                      |      |
| PUSHL  | R10                      | 8528 |
| PUSHL  | #1                       |      |
| PUSHL  | #164706                  |      |
| CALLS  | #3, LIB\$SIGNAL          |      |
| CMPB   | 6(R2), #2                | 8533 |
| BEQL   | 2\$                      |      |
| CMPB   | 6(R2), #3                | 8534 |
| BEQL   | 2\$                      |      |
| TSTL   | 8(R2)                    | 8536 |
| BNEQ   | 2\$                      |      |
| PUSHAB | P.AMJ                    | 8538 |
| PUSHL  | #1                       |      |
| PUSHL  | #164706                  |      |
| CALLS  | #3, LIB\$SIGNAL          |      |
| MOVL   | RIGHT_ARG, R4            | 8544 |
| CLRL   | R6                       |      |
| TSTL   | R4                       |      |
| BEQL   | 3\$                      |      |
| INCL   | R6                       |      |
| CMPB   | 6(R4), #2                | 8547 |
| BEQL   | 3\$                      |      |
| CMPB   | 6(R4), #3                | 8548 |
| BEQL   | 3\$                      |      |
| TSTL   | 8(R4)                    | 8550 |
| BNEQ   | 3\$                      |      |
| PUSHAB | P.AMK                    | 8552 |
| PUSHL  | #1                       |      |
| PUSHL  | #164706                  |      |
| CALLS  | #3, LIB\$SIGNAL          |      |
| TSTL   | R4                       | 8560 |
| BNEQ   | 4\$                      |      |
| TSTL   | RESULT                   |      |
| BNEQ   | 4\$                      |      |
| PUSHAB | P.AML                    | 8562 |

| Address | OpCode | OpName | OpType | OpValue | OpComment                   | OpStatus |
|---------|--------|--------|--------|---------|-----------------------------|----------|
| 0000    | 01     | DD     | 00084  |         | PUSHL #1                    |          |
| 0001    | 8F     | DD     | 00086  |         | PUSHL #164706               |          |
| 0002    | 03     | FB     | 0008C  |         | CALLS #3, LIB\$SIGNAL       |          |
| 0003    | 01     | DO     | 0008F  | 4\$:    | MOVL #1, STATUS             | 8567     |
| 0004    | AE     | D4     | 00092  |         | CLRL PARENT TYPE            | 8568     |
| 0005    | AC     | CF     | 00095  |         | CASEL TYPE INDEX, #1, #3    | 8569     |
| 0006    | 001A   |        | 0009A  | 5\$:    | .WORD 7\$-5\$,-             |          |
| 0007    |        |        |        |         | 18\$-5\$,-                  |          |
| 0008    |        |        |        |         | 39\$-5\$,-                  |          |
| 0009    |        |        |        |         | 29\$-5\$                    |          |
| 000A    | CA     | 9F     | 000A2  |         | PUSHAB P.AMM                | 8832     |
| 000B    | 01     | DD     | 000A6  |         | PUSHL #1                    |          |
| 000C    | 8F     | DD     | 000A8  |         | PUSHL #164706               |          |
| 000D    | 03     | FB     | 000AE  |         | CALLS #3, LIB\$SIGNAL       |          |
| 000E    | 02AD   | 31     | 000B1  | 6\$:    | BRW 39\$                    |          |
| 000F    | 56     | E9     | 000B4  | 7\$:    | BLBC R6, 11\$               |          |
| 0010    | A2     | D5     | 000B7  |         | TSTL 8(R2)                  | 8584     |
| 0011    | 10     | 13     | 000BA  |         | BEQL 8\$                    |          |
| 0012    | A4     | DD     | 000BC  |         | PUSHL 8(R4)                 | 8588     |
| 0013    | 08     | DD     | 000BF  |         | PUSHL 8(R2)                 | 8587     |
| 0014    | 08     | A2     | DD     | 000C2   | CALLS #2, TYPEID_CHECK_ENUM |          |
| 0015    | 02     | FB     | 000C7  |         | MOVL R0, STATUS             |          |
| 0016    | 50     | DO     | 000C7  |         | BRB 11\$                    | 8584     |
| 0017    | 18     | 11     | 000CA  |         | TSTL RESULT                 | 8593     |
| 0018    | AC     | D5     | 000CC  | 8\$:    | BNEQ 11\$                   |          |
| 0019    | 13     | 12     | 000CF  |         | CMPB 22(R2), #8             | 8596     |
| 001A    | A2     | 91     | 000D1  |         | BEQL 9\$                    |          |
| 001B    | 06     | 13     | 000D5  |         | CMPB 22(R2), #4             | 8597     |
| 001C    | A2     | 91     | 000D7  |         | BNEQ 10\$                   |          |
| 001D    | 05     | 12     | 000DB  |         | MOVL #1, STATUS             | 8599     |
| 001E    | 01     | DO     | 000DD  | 9\$:    | BRB 11\$                    |          |
| 001F    | 02     | 11     | 000E0  |         | CLRL STATUS                 | 8601     |
| 0020    | 55     | D4     | 000E2  | 10\$:   | MOVL RESULT, R3             | 8610     |
| 0021    | 53     | 10     | AC     | DO      | 000E4                       | 11\$:    |
| 0022    | C7     | 13     | 000E8  |         | BEQL 6\$                    |          |
| 0023    | 02     | 06     | A3     | 91      | 000EA                       |          |
| 0024    | C1     | 13     | 000EE  |         | CMPB 6(R3), #2              | 8611     |
| 0025    | 03     | 06     | A3     | 91      | 000F0                       |          |
| 0026    | BB     | 13     | 000F4  |         | BEQL 6\$                    | 8612     |
| 0027    | 55     | E9     | 000F6  |         | CMPB 6(R3), #3              |          |
| 0028    | 01     | 06     | A3     | 8F      | 000F9                       |          |
| 0029    | 014F   | 014F   | 000FE  | 12\$:   | BEQL 6\$                    | 8613     |
| 002A    | 014F   | 014F   | 00106  |         | BLBC STATUS, 6\$            | 8620     |
| 002B    | 014F   | 014F   | 0010E  |         | CASEB 6(R3), #1, #21        |          |
| 002C    | 014F   | 014F   | 00116  |         | .WORD 23\$-12\$,-           |          |
| 002D    | 014F   | 014F   | 0011E  |         | 23\$-12\$,-                 |          |
| 002E    | 014F   | 014F   | 00126  |         | 23\$-12\$,-                 |          |
| 002F    | 014F   | 014F   | 00126  |         | 23\$-12\$,-                 |          |
| 0030    | 014F   | 014F   | 00126  |         | 23\$-12\$,-                 |          |
| 0031    | 014F   | 014F   | 00126  |         | 23\$-12\$,-                 |          |
| 0032    | 014F   | 014F   | 00126  |         | 23\$-12\$,-                 |          |
| 0033    | 014F   | 014F   | 00126  |         | 23\$-12\$,-                 |          |
| 0034    | 014F   | 014F   | 00126  |         | 23\$-12\$,-                 |          |
| 0035    | 014F   | 014F   | 00126  |         | 23\$-12\$,-                 |          |
| 0036    | 014F   | 014F   | 00126  |         | 23\$-12\$,-                 |          |
| 0037    | 014F   | 014F   | 00126  |         | 23\$-12\$,-                 |          |
| 0038    | 014F   | 014F   | 00126  |         | 23\$-12\$,-                 |          |
| 0039    | 014F   | 014F   | 00126  |         | 23\$-12\$,-                 |          |
| 003A    | 014F   | 01     |        |         |                             |          |

| PC | Op | Op2 | Op3 | Op4 | Op5 | Op6 | Op7 | Op8 | Op9 | Op10 | Op11 | Op12 | Op13 | Op14 | Op15 | Op16 | Op17 | Op18 | Op19 | Op20 | Op21 | Op22 | Op23 | Op24 | Op25 | Op26 | Op27 | Op28 | Op29 | Op30 | Op31 | Op32 | Op33 | Op34 | Op35 | Op36 | Op37 | Op38 | Op39 | Op40 | Op41 | Op42 | Op43 | Op44 | Op45 | Op46 | Op47 | Op48 | Op49 | Op50 | Op51 | Op52 | Op53 | Op54 | Op55 | Op56 | Op57 | Op58 | Op59 | Op60 | Op61 | Op62 | Op63 | Op64 | Op65 | Op66 | Op67 | Op68 | Op69 | Op70 | Op71 | Op72 | Op73 | Op74 | Op75 | Op76 | Op77 | Op78 | Op79 | Op80 | Op81 | Op82 | Op83 | Op84 | Op85 | Op86 | Op87 | Op88 | Op89 | Op90 | Op91 | Op92 | Op93 | Op94 | Op95 | Op96 | Op97 | Op98 | Op99 | Op100 | Op101 | Op102 | Op103 | Op104 | Op105 | Op106 | Op107 | Op108 | Op109 | Op110 | Op111 | Op112 | Op113 | Op114 | Op115 | Op116 | Op117 | Op118 | Op119 | Op120 | Op121 | Op122 | Op123 | Op124 | Op125 | Op126 | Op127 | Op128 | Op129 | Op130 | Op131 | Op132 | Op133 | Op134 | Op135 | Op136 | Op137 | Op138 | Op139 | Op140 | Op141 | Op142 | Op143 | Op144 | Op145 | Op146 | Op147 | Op148 | Op149 | Op150 | Op151 | Op152 | Op153 | Op154 | Op155 | Op156 | Op157 | Op158 | Op159 | Op160 | Op161 | Op162 | Op163 | Op164 | Op165 | Op166 | Op167 | Op168 | Op169 | Op170 | Op171 | Op172 | Op173 | Op174 | Op175 | Op176 | Op177 | Op178 | Op179 | Op180 | Op181 | Op182 | Op183 | Op184 | Op185 | Op186 | Op187 | Op188 | Op189 | Op190 | Op191 | Op192 | Op193 | Op194 | Op195 | Op196 | Op197 | Op198 | Op199 | Op200 | Op201 | Op202 | Op203 | Op204 | Op205 | Op206 | Op207 | Op208 | Op209 | Op210 | Op211 | Op212 | Op213 | Op214 | Op215 | Op216 | Op217 | Op218 | Op219 | Op220 | Op221 | Op222 | Op223 | Op224 | Op225 | Op226 | Op227 | Op228 | Op229 | Op230 | Op231 | Op232 | Op233 | Op234 | Op235 | Op236 | Op237 | Op238 | Op239 | Op240 | Op241 | Op242 | Op243 | Op244 | Op245 | Op246 | Op247 | Op248 | Op249 | Op250 | Op251 | Op252 | Op253 | Op254 | Op255 | Op256 | Op257 | Op258 | Op259 | Op260 | Op261 | Op262 | Op263 | Op264 | Op265 | Op266 | Op267 | Op268 | Op269 | Op270 | Op271 | Op272 | Op273 | Op274 | Op275 | Op276 | Op277 | Op278 | Op279 | Op280 | Op281 | Op282 | Op283 | Op284 | Op285 | Op286 | Op287 | Op288 | Op289 | Op290 | Op291 | Op292 | Op293 | Op294 | Op295 | Op296 | Op297 | Op298 | Op299 | Op300 | Op301 | Op302 | Op303 | Op304 | Op305 | Op306 | Op307 | Op308 | Op309 | Op310 | Op311 | Op312 | Op313 | Op314 | Op315 | Op316 | Op317 | Op318 | Op319 | Op320 | Op321 | Op322 | Op323 | Op324 | Op325 | Op326 | Op327 | Op328 | Op329 | Op330 | Op331 | Op332 | Op333 | Op334 | Op335 | Op336 | Op337 | Op338 | Op339 | Op340 | Op341 | Op342 | Op343 | Op344 | Op345 | Op346 | Op347 | Op348 | Op349 | Op350 | Op351 | Op352 | Op353 | Op354 | Op355 | Op356 | Op357 | Op358 | Op359 | Op360 | Op361 | Op362 | Op363 | Op364 | Op365 | Op366 | Op367 | Op368 | Op369 | Op370 | Op371 | Op372 | Op373 | Op374 | Op375 | Op376 | Op377 | Op378 | Op379 | Op380 | Op381 | Op382 | Op383 | Op384 | Op385 | Op386 | Op387 | Op388 | Op389 | Op390 | Op391 | Op392 | Op393 | Op394 | Op395 | Op396 | Op397 | Op398 | Op399 | Op400 | Op401 | Op402 | Op403 | Op404 | Op405 | Op406 | Op407 | Op408 | Op409 | Op410 | Op411 | Op412 | Op413 | Op414 | Op415 | Op416 | Op417 | Op418 | Op419 |
|----|----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|----|----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|

| Address | Instruction | Op | Op2 | Op3 | Op4 | Op5 | Op6 | Op7 | Op8 | Op9 | Op10 | Op11 | Op12 | Op13 | Op14 | Op15 | Op16 | Op17 | Op18 | Op19 | Op20 | Op21 | Op22 | Op23 | Op24 | Op25 | Op26 | Op27 | Op28 | Op29 | Op30 | Op31 | Op32 | Op33 | Op34 | Op35 | Op36 | Op37 | Op38 | Op39 | Op40 | Op41 | Op42 | Op43 | Op44 | Op45 | Op46 | Op47 | Op48 | Op49 | Op50 | Op51 | Op52 | Op53 | Op54 | Op55 | Op56 | Op57 | Op58 | Op59 | Op60 | Op61 | Op62 | Op63 | Op64 | Op65 | Op66 | Op67 | Op68 | Op69 | Op70 | Op71 | Op72 | Op73 | Op74 | Op75 | Op76 | Op77 | Op78 | Op79 | Op80 | Op81 | Op82 | Op83 | Op84 | Op85 | Op86 | Op87 | Op88 | Op89 | Op90 | Op91 | Op92 | Op93 | Op94 | Op95 | Op96 | Op97 | Op98 | Op99 | Op100 | Op101 | Op102 | Op103 | Op104 | Op105 | Op106 | Op107 | Op108 | Op109 | Op110 | Op111 | Op112 | Op113 | Op114 | Op115 | Op116 | Op117 | Op118 | Op119 | Op120 | Op121 | Op122 | Op123 | Op124 | Op125 | Op126 | Op127 | Op128 | Op129 | Op130 | Op131 | Op132 | Op133 | Op134 | Op135 | Op136 | Op137 | Op138 | Op139 | Op140 | Op141 | Op142 | Op143 | Op144 | Op145 | Op146 | Op147 | Op148 | Op149 | Op150 | Op151 | Op152 | Op153 | Op154 | Op155 | Op156 | Op157 | Op158 | Op159 | Op160 | Op161 | Op162 | Op163 | Op164 | Op165 | Op166 | Op167 | Op168 | Op169 | Op170 | Op171 | Op172 | Op173 | Op174 | Op175 | Op176 | Op177 | Op178 | Op179 | Op180 | Op181 | Op182 | Op183 | Op184 | Op185 | Op186 | Op187 | Op188 | Op189 | Op190 | Op191 | Op192 | Op193 | Op194 | Op195 | Op196 | Op197 | Op198 | Op199 | Op200 | Op201 | Op202 | Op203 | Op204 | Op205 | Op206 | Op207 | Op208 | Op209 | Op210 | Op211 | Op212 | Op213 | Op214 | Op215 | Op216 | Op217 | Op218 | Op219 | Op220 | Op221 | Op222 | Op223 | Op224 | Op225 | Op226 | Op227 | Op228 | Op229 | Op230 | Op231 | Op232 | Op233 | Op234 | Op235 | Op236 | Op237 | Op238 | Op239 | Op240 | Op241 | Op242 | Op243 | Op244 | Op245 | Op246 | Op247 | Op248 | Op249 | Op250 | Op251 | Op252 | Op253 | Op254 | Op255 | Op256 | Op257 | Op258 | Op259 | Op260 | Op261 | Op262 | Op263 | Op264 | Op265 | Op266 | Op267 | Op268 | Op269 | Op270 | Op271 | Op272 | Op273 | Op274 | Op275 | Op276 | Op277 | Op278 | Op279 | Op280 | Op281 | Op282 | Op283 | Op284 | Op285 | Op286 | Op287 | Op288 | Op289 | Op290 | Op291 | Op292 | Op293 | Op294 | Op295 | Op296 | Op297 | Op298 | Op299 | Op300 | Op301 | Op302 | Op303 | Op304 | Op305 | Op306 | Op307 | Op308 | Op309 | Op310 | Op311 | Op312 | Op313 | Op314 | Op315 | Op316 | Op317 | Op318 | Op319 | Op320 | Op321 | Op322 | Op323 | Op324 | Op325 | Op326 | Op327 | Op328 | Op329 | Op330 | Op331 | Op332 | Op333 | Op334 | Op335 | Op336 | Op337 | Op338 | Op339 | Op340 | Op341 | Op342 | Op343 | Op344 | Op345 | Op346 | Op347 | Op348 | Op349 | Op350 | Op351 | Op352 | Op353 | Op354 | Op355 | Op356 | Op357 | Op358 | Op359 | Op360 | Op361 | Op362 | Op363 | Op364 | Op365 | Op366 | Op367 | Op368 | Op369 | Op370 | Op371 | Op372 | Op373 | Op374 | Op375 | Op376 | Op377 | Op378 | Op379 | Op380 | Op381 | Op382 | Op383 | Op384 | Op385 | Op386 | Op387 | Op388 | Op389 | Op390 | Op391 | Op392 | Op393 | Op394 | Op395 | Op396 | Op397 | Op398 | Op399 | Op400 | Op401 | Op402 | Op403 | Op404 | Op405 | Op406 | Op407 | Op408 | Op409 | Op410 | Op411 | Op412 | Op413 | Op414 | Op415 | Op416 | Op417 | Op418 |
|---------|-------------|----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|---------|-------------|----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|

|           |    |      |      |    |       |        |                                 |      |
|-----------|----|------|------|----|-------|--------|---------------------------------|------|
|           | 53 | 18   | A4   | 9A | 00273 | MOVZBL | 24(RESULT_TYPEID), RESULT_FCODE | 8721 |
|           | 02 |      | 53   | D1 | 00277 | CMPL   | RESULT_FCODE, #2                | 8722 |
|           |    |      | 19   | 12 | 0027A | BNEQ   | 27\$                            |      |
|           |    | 14   | AE   | 9F | 0027C | PUSHAB | SIZE                            | 8725 |
|           |    | 1C   | AE   | 9F | 0027F | PUSHAB | RESULT_DTYPE                    |      |
|           |    |      | 54   | DD | 00282 | PUSHL  | RESULT_TYPEID                   |      |
|           | 6B |      | 03   | FB | 00284 | CALLS  | #3, DBG\$STA_TYP_ATOMIC         |      |
| 0000009E  | 8F | 18   | AE   | D1 | 00287 | CMPL   | RESULT_DTYPE, #T58              | 8726 |
|           |    |      | 04   | 12 | 0028F | BNEQ   | 27\$                            |      |
| 18        | AE |      | 28   | DD | 00291 | MOVL   | #40, RESULT_DTYPE               |      |
|           |    | 18   | AE   | DD | 00295 | PUSHL  | RESULT_DTYPE                    | 8731 |
|           |    | 04   | AE   | DD | 00298 | PUSHL  | LEFT_DTYPE                      |      |
|           |    |      | 53   | DD | 0029B | PUSHL  | RESULT_FCODE                    | 8730 |
|           |    | 0110 | 8F   | BB | 0029D | PUSHR  | #*M<R4,R8>                      | 8729 |
|           |    |      | 57   | DD | 002A1 | PUSHL  | LEFT_TYPEID                     |      |
| 0000V     | CF |      | 06   | FB | 002A3 | CALLS  | #6, TYPEID_CHECK_SET            |      |
|           | 55 |      | 50   | DD | 002AB | MOVL   | R0, STATUS                      |      |
|           | A1 |      | 55   | E9 | 002AB | BLBC   | STATUS, 24\$                    | 8733 |
|           | 50 | 1C   | AE   | DD | 002AE | MOVL   | PARENT_TYPE, R0                 | 8739 |
|           |    |      | 09   | 13 | 002B2 | BEQL   | 28\$                            |      |
|           | 09 | 18   | A0   | 91 | 002B4 | CMPB   | 24(R0), #9                      | 8741 |
|           |    |      | 03   | 12 | 002B8 | BNEQ   | 28\$                            |      |
|           | 54 |      | 50   | DD | 002BA | MOVL   | R0, RESULT_TYPEID               | 8743 |
|           | 50 | 18   | A4   | 9A | 002BD | MOVZBL | 24(RESULT_TYPEID), R0           | 8748 |
|           | 09 |      | 50   | 91 | 002C1 | CMPB   | R0, #9                          | 8750 |
|           |    |      | 89   | 12 | 002C4 | BNEQ   | 24\$                            |      |
|           |    |      | 54   | DD | 002C6 | PUSHL  | RESULT_TYPEID                   | 8751 |
|           |    |      | 008E | 31 | 002C8 | BRW    | 38\$                            |      |
|           | 50 | 10   | AC   | DD | 002CB | MOVL   | RESULT, R0                      | 8781 |
|           |    |      | 03   | 13 | 002CF | BEQL   | 30\$                            |      |
|           |    |      | 0082 | 31 | 002D1 | BRW    | 37\$                            |      |
| 10        | AE | 08   | A4   | DD | 002D4 | MOVL   | 8(R4), PARENT_TYPE              | 8784 |
|           | 50 | 10   | AE   | DD | 002D9 | MOVL   | PARENT_TYPE, R0                 | 8785 |
|           | 09 | 18   | A0   | 91 | 002DD | CMPB   | 24(R0), #9                      |      |
|           |    |      | 17   | 12 | 002E1 | BNEQ   | 32\$                            |      |
|           |    | 14   | AE   | 9F | 002E3 | PUSHAB | SIZE                            | 8786 |
|           |    | 0C   | AE   | 9F | 002E6 | PUSHAB | DUMMY2                          |      |
|           |    | 14   | AE   | 9F | 002E9 | PUSHAB | DUMMY1                          |      |
|           |    | 1C   | AE   | 9F | 002EC | PUSHAB | PARENT_TYPE                     |      |
|           |    |      | 50   | DD | 002EF | PUSHL  | R0                              |      |
| 00000000G | 00 |      | 05   | FB | 002F1 | CALLS  | #5, DBG\$STA_TYP_SUBRNG         |      |
|           |    |      | DF   | 11 | 002F8 | BRB    | 31\$                            |      |
|           | 50 | 10   | AE   | DD | 002FA | MOVL   | PARENT_TYPE, R0                 | 8788 |
|           | 51 | 18   | A0   | 9A | 002FE | MOVZBL | 24(R0), R1                      |      |
|           | 04 |      | 51   | 91 | 00302 | CMPB   | R1, #4                          | 8790 |
|           |    |      | 23   | 12 | 00305 | BNEQ   | 35\$                            |      |
|           |    | 08   | A2   | D5 | 00307 | TSTL   | 8(R2)                           | 8792 |
|           |    |      | 0F   | 13 | 0030A | BEQL   | 33\$                            |      |
|           |    |      | 50   | DD | 0030C | PUSHL  | R0                              | 8796 |
|           |    | 08   | A2   | DD | 0030E | PUSHL  | 8(R2)                           | 8795 |
| 0000V     | CF |      | 02   | FB | 00311 | CALLS  | #2, TYPEID_CHECK_ENUM           |      |
|           | 55 |      | 50   | DD | 00316 | MOVL   | R0, STATUS                      |      |
|           |    |      | 46   | 11 | 00319 | BRB    | 39\$                            | 8792 |
|           | 08 | 16   | A2   | 91 | 0031B | CMPB   | 22(R2), #8                      | 8801 |
|           |    |      | 40   | 13 | 0031F | BEQL   | 39\$                            |      |
|           | 04 | 16   | A2   | 91 | 00321 | CMPB   | 22(R2), #4                      | 8802 |
|           |    |      | 3A   | 13 | 00325 | BEQL   | 39\$                            |      |

DBGVALOP  
V04-000

D 15  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGVALOP.B32;1

Page 260  
(37)

|    |    |          |    |    |      |    |       |       |        |                               |  |      |
|----|----|----------|----|----|------|----|-------|-------|--------|-------------------------------|--|------|
|    |    |          |    |    | FF23 | 31 | 00327 | 34\$: | BRW    | 23\$                          |  | 8804 |
|    | 02 |          |    |    | 51   | 91 | 0032A | 35\$: | CMPB   | R1, #2                        |  | 8808 |
|    |    |          |    |    | F8   | 12 | 0032D |       | BNEQ   | 34\$                          |  |      |
|    |    |          |    |    | 14   | AE | 9F    | 0032F | PUSHAB | SIZE                          |  | 8810 |
|    |    |          |    |    | 1C   | AE | 9F    | 00332 | PUSHAB | RESULT_DTYPE                  |  |      |
|    |    |          |    |    |      | 50 | DD    | 00335 | PUSHL  | R0                            |  |      |
|    |    |          |    |    |      | 03 | F8    | 00337 | CALLS  | #3, DBG\$STA_TYP_ATOMIC       |  |      |
|    |    | 0000009E |    | 6B |      |    |       |       | CMPL   | RESULT_DTYPE, #T58            |  | 8811 |
|    |    |          |    | 8F | 18   | AE | D1    | 0033A | BNEQ   | 36\$                          |  |      |
|    |    |          |    |    |      | 04 | 12    | 00342 | MOVL   | #40, RESULT_DTYPE             |  |      |
|    |    |          |    | 18 | AE   | 28 | D0    | 00344 | CMPZV  | #0, #8, 22(R2), RESULT_DTYPE  |  | 8812 |
| 18 | AE |          | 16 | A2 |      | 00 | ED    | 00348 | BNEQ   | 34\$                          |  |      |
|    |    |          |    |    |      | D6 | 12    | 0034F | MOVL   | #1, STATUS                    |  | 8814 |
|    |    |          |    |    |      | 55 | 01    | D0    | BRB    | 39\$                          |  |      |
|    |    |          |    |    |      |    | 08    | 11    | PUSHL  | 8(R0)                         |  | 8828 |
|    |    |          |    |    | 08   | A0 | DD    | 00356 | PUSHL  | R2                            |  |      |
|    |    |          |    |    |      | 52 | DD    | 00359 | CALLS  | #2, TYPEID_RANGE_CHECK_SUBRNG |  |      |
|    |    | 0000V    |    | CF |      | 02 | F8    | 0035B | RET    |                               |  |      |
|    |    |          |    |    |      |    | 04    | 00360 | MOVL   | STATUS, R0                    |  | 8835 |
|    |    |          |    |    |      | 55 | D0    | 00361 | RET    |                               |  | 8836 |
|    |    |          |    |    |      |    | 04    | 00364 |        |                               |  |      |

; Routine Size: 869 bytes, Routine Base: DBG\$CODE + 1954

```
8741 8837 1 ROUTINE DBG$TYPE_CONV (VALUE1, VALUE2) =
8742 8838 1
8743 8839 1 FUNCTION
8744 8840 1     This routine is the top-level type converter for DEBUG. It takes a
8745 8841 1     source value descriptor and a target value descriptor. It first
8746 8842 1     checks whether there are any language-specific type conversion
8747 8843 1     rules to use. If not, it calls the language-independent routine
8748 8844 1     DBG$COVER_DX_DX.
8749 8845 1
8750 8846 1 INPUTS
8751 8847 1     VALUE1 - Pointer to a value descriptor to be type-converted.
8752 8848 1
8753 8849 1     VALUE2 - Pointer to the target value descriptor.
8754 8850 1
8755 8851 1 OUTPUTS
8756 8852 1     A pointer to a value descriptor is returned. The target descriptor
8757 8853 1     is filled in with the result of the conversion.
8758 8854 1
8759 8855 2 BEGIN
8760 8856 2
8761 8857 2 MAP
8762 8858 2     VALUE1: REF DBG$VALDESC,
8763 8859 2     VALUE2: REF DBG$VALDESC;
8764 8860 2
8765 8861 2 LOCAL
8766 8862 2     CVT_TBL_ENTRY: TYPE CVT$ENTRY, ! An entry in the Type Conversion Table
8767 8863 2     CVT_TYPE_PAIR: TYPE$PAIR,      ! Data Type Pairs
8768 8864 2     STATUS;                        ! Return status from lib$cvr_dx_dx
8769 8865 2
8770 8866 2
8771 8867 2 ! Perform Type Conversion indicated by Language Dependent Type Conversion
8772 8868 2 ! Table.
8773 8869 2 ! (Left type in Type Pair = The type we want to convert into,
8774 8870 2 ! Right type = From type)
8775 8871 2
8776 8872 2 IF .CVT_TBL NEQ TABLEBASE
8777 8873 2 THEN
8778 8874 2 BEGIN
8779 8875 2     CVT_TYPE_PAIR[BS_LEFT_TYPE] = .VALUE2[DBG$B_VALUE_DTYPE];
8780 8876 2     CVT_TYPE_PAIR[BS_RIGHT_TYPE] = .VALUE1[DBG$B_VALUE_DTYPE];
8781 8877 2     INCR I FROM 0 TO .CVT_TBL_SIZE - 1 DO
8782 8878 2         BEGIN
8783 8879 2             CVT_TBL_ENTRY = .CVT_TBL[I];
8784 8880 2             IF .CVT_TBL_ENTRY EQ 0 THEN EXITLOOP;
8785 8881 2             IF (.CVT_TBL_ENTRY[TYPE_CVT$B_LOWER_TYPE] EQL DSC$K_DTYPE_ANY AND
8786 8882 2                 .CVT_TBL_ENTRY[TYPE_CVT$B_HIGHER_TYPE] EQL .CVT_TYPE_PAIR[BS_RIGHT_TYPE]) OR
8787 8883 2                 (.CVT_TBL_ENTRY[TYPE_CVT$B_HIGHER_TYPE] EQL DSC$K_DTYPE_ANY AND
8788 8884 2                 .CVT_TBL_ENTRY[TYPE_CVT$B_LOWER_TYPE] EQL .CVT_TYPE_PAIR[BS_LEFT_TYPE]) OR
8789 8885 2                 (.CVT_TBL_ENTRY[TYPE_CVT$W_MAP_PAIR] EQL .CVT_TYPE_PAIR[WS_TYPE_PAIR])
8790 8886 2             THEN
8791 8887 2                 BEGIN
8792 8888 2                     IF .CVT_TBL_ENTRY[TYPE_CVT$W_ROUT] NEQ 0
8793 8889 2                     THEN
8794 8890 2                         DBG$LANGUAGE_TYPE_CONV (.CVT_TBL_ENTRY[TYPE_CVT$W_ROUT],
8795 8891 2                         .VALUE1, .VALUE2);
8796 8892 2                 RETURN .VALUE2;
8797 8893 2             END;
```

8798  
8799  
8800  
8801  
8802  
8803  
8804  
8805  
8806  
8807  
8808  
8809  
8810  
8811  
8812  
8813

8894  
8895  
8896  
8897  
8898  
8899  
8900  
8901  
8902  
8903  
8904  
8905  
8906  
8907  
8908  
8909

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176

DBGVALOP  
V04-000

G 15

16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGVALOP.B32;1

Page 263  
(38)

|           |    |    |    |          |      |       |                      |   |      |
|-----------|----|----|----|----------|------|-------|----------------------|---|------|
| 50        | 08 | AC | D0 | 0007E    | 7\$: | MOVL  | VALUE2, R0           | : | 8901 |
| 30        | 16 | A0 | 91 | 00082    |      | CMPB  | 22(R0), #48          | : |      |
|           |    | 0A | 12 | 00086    |      | BNEQ  | 8\$                  | : |      |
| 05        | 06 | A0 | 91 | 00088    |      | CMPB  | 6(R0), #5            | : | 8902 |
|           |    | 04 | 12 | 0008C    |      | BNEQ  | 8\$                  | : |      |
| 16        | A0 | 0E | 90 | 0008E    |      | MOVB  | #14, 22(R0)          | : | 8904 |
|           |    | FC | A8 | DD 00092 | 8\$: | PUSHL | CVT_ROUND_FLAG       | : | 8906 |
|           |    | 50 | DD | 00095    |      | PUSHL | R0                   | : |      |
|           |    | 04 | AC | DD 00097 |      | PUSHL | VALUE1               | : |      |
| 00000000G | 00 | 03 | FB | 0009A    |      | CALLS | #3, DBG\$COVER_DX_DX | : |      |
| 08        | AC | 50 | D0 | 000A1    |      | MOVL  | R0, VALUE2           | : |      |
|           |    |    | 04 | 000A5    |      | RET   |                      | : | 8909 |

; Routine Size: 166 bytes, Routine Base: DBG\$CODE + 1CB9

```
8815 8910 1 ROUTINE CONV_TEXT_PACK_VALUE (VAL_DESC) =
8816 8911 1
8817 8912 1 FUNCTION
8818 8913 1     This routine is used to convert the text string to pack decimal.
8819 8914 1
8820 8915 1 INPUTS
8821 8916 1     VAL_DESC      - Unconverted constant.
8822 8917 1
8823 8918 1 OUTPUTS
8824 8919 1     A converted value descriptor is returned.
8825 8920 1
8826 8921 1
8827 8922 1 BEGIN
8828 8923 1
8829 8924 1 MAP
8830 8925 1     VAL_DESC: REF DBG$VALDESC;
8831 8926 1
8832 8927 1
8833 8928 1 LOCAL
8834 8929 1     DOT: REF VECTOR[BYTE],
8835 8930 1     END_PTR: REF VECTOR[BYTE],
8836 8931 1     P_VAL_DESC: REF DBG$VALDESC,
8837 8932 1     SCALE;
8838 8933 1
8839 8934 1
8840 8935 1 P_VAL_DESC = MAKE_VAL_DESC(DSC$K_DTYPE_P,
8841 8936 1     DBG$NOM_BYTES(DSC$K_DTYPE_P),
8842 8937 1     0,
8843 8938 1     TRUE);
8844 8939 1
8845 8940 1 DOT = CH$FIND_CH(.VAL_DESC[DBG$W_VALUE_LENGTH],
8846 8941 1     .VAL_DESC[DBG$L_VALUE_POINTER],
8847 8942 1     %C'.');
8848 8943 1
8849 8944 1
8850 8945 1 ! A string of digits has no '.'.
8851 8946 1
8852 8947 1 IF .DOT EQL 0
8853 8948 1 THEN
8854 8949 1 BEGIN
8855 8950 1     SCALE = 0;
8856 8951 1     P_VAL_DESC[DBG$W_VALUE_LENGTH] = .VAL_DESC[DBG$W_VALUE_LENGTH];
8857 8952 1     P_VAL_DESC[DBG$B_VALUE_DIGITS] = .VAL_DESC[DBG$W_VALUE_LENGTH];
8858 8953 1 END
8859 8954 1
8860 8955 1
8861 8956 1 ! A string of digits has '.'.
8862 8957 1
8863 8958 1 ELSE
8864 8959 1 BEGIN
8865 8960 1     END_PTR = .VAL_DESC[DBG$L_VALUE_POINTER] + .VAL_DESC[DBG$W_VALUE_LENGTH] - 1;
8866 8961 1     SCALE = .END_PTR - .DOT;
8867 8962 1
8868 8963 1
8869 8964 1 ! Fill in the digits and fill in scaling factor after the
8870 8965 1 ! type conversion.
8871 8966 1
```

! Convert the string to decimal.

|    |       |    |    | 03FC 00000 | CONV_TEXT | PACK_VALUE: |        |                                |      |
|----|-------|----|----|------------|-----------|-------------|--------|--------------------------------|------|
|    |       |    |    | 01         | DD        | 00002       | .WORD  | Save R2,R3,R4,R5,R6,R7,R8,R9   | 8910 |
|    |       |    |    | 15         | 7D        | 00004       | PUSHL  | #1                             | 8935 |
|    | FB23  | 7E |    | 01         | FB        | 00007       | MOVQ   | #21, -(SP)                     | 8936 |
|    |       | CF |    | 50         | DD        | 0000C       | CALLS  | #1, DBGSNUM_BYTES              |      |
|    |       |    |    | 15         | DD        | 0000E       | PUSHL  | R0                             |      |
|    | 0000V | CF |    | 04         | FB        | 00010       | PUSHL  | #21                            | 8935 |
|    |       | 58 |    | 50         | D0        | 00015       | CALLS  | #4, MAKE_VAL_DESC              |      |
|    |       | 57 | 04 | 50         | D0        | 00015       | MOVL   | R0, P_VAL_DESC                 |      |
|    |       | 52 | 14 | AC         | D0        | 00018       | MOVL   | VAL_DESC, -R7                  | 8940 |
| 18 | B7    | 52 |    | A7         | 3C        | 0001C       | MOVZWL | 20(R7), R2                     |      |
|    |       |    |    | 2E         | 3A        | 00020       | LOCC   | #46, R2, @24(R7)               |      |
|    |       |    |    | 02         | 12        | 00025       | BNEQ   | 1\$                            |      |
|    |       |    |    | 51         | D4        | 00027       | CLRL   | R1                             |      |
|    |       |    |    | 51         | D5        | 00029       | TSTL   | DOT                            | 8947 |
|    |       |    |    | 0C         | 12        | 0002B       | BNEQ   | 2\$                            |      |
|    |       |    |    | 59         | D4        | 0002D       | CLRL   | SCALE                          | 8950 |
|    | 14    | A8 |    | 52         | B0        | 0002F       | MOVW   | R2, 20(P_VAL_DESC)             | 8951 |
|    | 1D    | A8 |    | 52         | 90        | 00033       | MOVB   | R2, 29(P_VAL_DESC)             | 8952 |
|    |       |    |    | 26         | 11        | 00037       | BRB    | 4\$                            | 8947 |
| 56 |       | 52 | 18 | A7         | C1        | 00039       | ADDL3  | 24(R7), R2, R6                 | 8960 |
|    |       |    |    | 56         | D7        | 0003E       | DECL   | END_PTR                        |      |
|    | 59    | 56 |    | 51         | C3        | 00040       | SUBL3  | DOT, END_PTR, SCALE            | 8961 |
| 14 | A8    | 52 |    | 01         | A3        | 00044       | SUBW3  | #1, R2, 20(P_VAL_DESC)         | 8967 |
|    |       | A8 | 14 | A8         | 90        | 00049       | MOVB   | 20(P_VAL_DESC), 29(P_VAL_DESC) | 8968 |
|    |       | 56 |    | 51         | D1        | 0004E       | CPL    | DOT, END_PTR                   | 8973 |
|    |       |    |    | 05         | 18        | 00051       | BGEQ   | 3\$                            |      |
| 61 | 01    | A1 |    | 59         | 28        | 00053       | MOVC3  | SCALE, 1(DOT), (DOT)           | 8975 |
|    |       |    |    | 56         | D4        | 00058       | CLRL   | END_PTR                        | 8977 |

DBGEVALOP  
V04-000

J 15  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32;1

Page 266  
(39)

|      |    |    |    |       |       |          |                        |   |      |
|------|----|----|----|-------|-------|----------|------------------------|---|------|
| 14   | A7 | 14 | A8 | 80    | 0005A | MOVW     | 20(P_VAL_DESC), 20(R7) | : | 8978 |
| 16   | A7 |    | 0E | 90    | 0005F | 48: MOVB | #14, -22(R7)           | : | 8984 |
|      |    | 10 | A7 | B4    | 00063 | CLRW     | 16(R7)                 | : | 8985 |
|      | 7E |    | 57 | 7D    | 00066 | MOVQ     | R7, -(SP)              | : | 8986 |
| FEEC | CF |    | 02 | FB    | 00069 | CALLS    | #2, DBG\$TYPE_CONV     | : |      |
|      | 58 |    | 50 | D0    | 0006E | MOVL     | R0, P_VAL_DESC         | : |      |
| 1C   | A8 |    | 59 | 8E    | 00071 | MNEGB    | SCALE, 28(P_VAL_DESC)  | : | 8987 |
|      | 50 |    | 58 | D0    | 00075 | MOVL     | P_VAL_DESC, -R0        | : | 8988 |
|      |    |    | 04 | 00078 |       | RET      |                        | : | 8989 |

; Routine Size: 121 bytes, Routine Base: DBG\$CODE + 1D5F

; 8895 8990 1

```
8897 8991 1 ROUTINE GET_DATA_LENGTH(FROM_DTYPE, TO_DTYPE, FROM_LENGTH) =
8898 8992 1
8899 8993 1 FUNCTION
8900 8994 1     This routine tries to guess the length of the TO_DTYPE given
8901 8995 1     FROM_DTYPE. Sometimes, it is not obvious what is the length
8902 8996 1     for the TO_DTYPE, for example, L --> V. This routine first
8903 8997 1     will fill in the normal length, then will correct the length
8904 8998 1     by the language rules. Note: language code in here is not
8905 8999 1     tested for most of the cases are isolated.
8906 9000 1
8907 9001 1 INPUTS
8908 9002 1     FROM_DTYPE      - Source dtype.
8909 9003 1
8910 9004 1     TO_DTYPE        - Target dtype.
8911 9005 1
8912 9006 1     FROM_LENGTH     - Source dtype length.
8913 9007 1
8914 9008 1 OUTPUTS
8915 9009 1     TO_LENGTH (target dtype length) is returned.
8916 9010 1
8917 9011 1
8918 9012 1 BEGIN
8919 9013 1
8920 9014 1 LOCAL
8921 9015 1     TO_LENGTH;
8922 9016 1
8923 9017 1
8924 9018 1     ! Fill in the length by normal way. Just in case, we did not update
8925 9019 1     ! the length correctly later on, we still have a length.
8926 9020 1
8927 9021 1     TO_LENGTH = DBG$NUM_BYTES(.TO_DTYPE);
8928 9022 1
8929 9023 1
8930 9024 1     ! Update the length by specified rules.
8931 9025 1
8932 9026 1     CASE .TO_DTYPE FROM DBG$K_MINIMUM_DTYPE TO DBG$K_MAXIMUM_DTYPE OF
8933 9027 1     SET
8934 9028 1
8935 9029 1
8936 9030 1     ! If we are converting to FIXED then we use the length from
8937 9031 1     ! the source descriptor.
8938 9032 1
8939 9033 1     [DSC$K_DTYPE_FIXED]:
8940 9034 1         TO_LENGTH = .FROM_LENGTH;
8941 9035 1
8942 9036 1
8943 9037 1     [DSC$K_DTYPE_V, DSC$K_DTYPE_VU]:
8944 9038 1         BEGIN
8945 9039 1             CASE .FROM_DTYPE FROM DSC$K_DTYPE_LOWEST TO DSC$K_DTYPE_HIGHEST OF
8946 9040 1             SET
8947 9041 1                 [DSC$K_DTYPE_V, DSC$K_DTYPE_VU]:
8948 9042 1                     TO_LENGTH = .FROM_LENGTH;
8949 9043 1                 [DSC$K_DTYPE_B]:
8950 9044 1                     TO_LENGTH = 7;
8951 9045 1                 [DSC$K_DTYPE_W]:
8952 9046 1                     TO_LENGTH = 15;
8953 9047 1                 [DSC$K_DTYPE_P]:
```

```

8954      9048      4      BEGIN
8955      9049      4      TO_LENGTH = ((.FROM_LENGTH * 332) + 92) / 100;
8956      9050      4      TO_LENGTH = MIN(31, .TO_LENGTH);
8957      9051      4      END;
8958      9052      4      [INRANGE]:
8959      9053      4      TO_LENGTH = 31;
8960      9054      4      [OUTRANGE]:
8961      9055      4      BEGIN
8962      9056      4      IF .FROM_DTYPE EQL DBG$K_DTYPE_PICT
8963      9057      4      THEN
8964      9058      4      TO_LENGTH = 31
8965      9059      4      ELSE
8966      9060      4      $DBG_ERROR('DBGEVALOP\GET_DATA_LENGTH');
8967      9061      4      END;
8968      9062      4      TES;
8969      9063      4      END;
8970      9064      4      [DSC$K_DTYPE_T, DSC$K_DTYPE_VT]:
8971      9065      4      BEGIN
8972      9066      4      CASE .FROM_DTYPE FROM DSC$K_DTYPE_LOWEST TO DSC$K_DTYPE_HIGHEST OF
8973      9067      4      SET
8974      9068      4      [DSC$K_DTYPE_V, DSC$K_DTYPE_VU, DSC$K_DTYPE_T, DSC$K_DTYPE_VT]:
8975      9069      4      TO_LENGTH = .FROM_LENGTH;
8976      9070      4      [DSC$K_DTYPE_B]:
8977      9071      4      TO_LENGTH = 7;
8978      9072      4      [DSC$K_DTYPE_W]:
8979      9073      4      TO_LENGTH = 7;
8980      9074      4      [DSC$K_DTYPE_L]:
8981      9075      4      TO_LENGTH = 15;
8982      9076      4      [DSC$K_DTYPE_F]:
8983      9077      4      TO_LENGTH = 15;
8984      9078      4      [DSC$K_DTYPE_D, DSC$K_DTYPE_G]:
8985      9079      4      TO_LENGTH = 25;
8986      9080      4      [DSC$K_DTYPE_H]:
8987      9081      4      TO_LENGTH = 42;
8988      9082      4      [DSC$K_DTYPE_P]:
8989      9083      4      TO_LENGTH = .FROM_LENGTH + 5;
8990      9084      4      [INRANGE, OUTRANGE]:
8991      9085      4      BEGIN
8992      9086      4      IF .FROM_DTYPE EQL DBG$K_DTYPE_PICT
8993      9087      4      THEN
8994      9088      4      TO_LENGTH = .FROM_LENGTH
8995      9089      4      ELSE
8996      9090      4      $DBG_ERROR('DBGEVALOP\GET_DATA_LENGTH');
8997      9091      4      END;
8998      9092      4      TES;
8999      9093      4      END;
9000      9094      4      [INRANGE, OUTRANGE]:
9001      9095      4      BEGIN
9002      9096      4      0;
9003      9097      4      END;
9004      9098      4      TES;
9005      9099      4      RETURN .TO_LENGTH;
9006      9100      4      END;
9007      9101      4
9008      9102      4
9009      9103      4
9010      9104      4
```

```
.PSECT DBGSPLIT,NOWRT, SHR, PIC,0
```

1

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

0004 00000 GET\_DATA\_LENGTH:

8991  
9021  
9026

007E  
007E  
007E  
007E  
007E  
007E  
007E  
007E  
007E  
007E

24  
007E  
00FA  
007E  
007E  
007E  
007E  
007E  
007E  
007E  
007E

01  
007E  
00F5  
007E  
007E  
007E  
007E  
007E  
007E  
007E  
00EF

04 0113 31 0006A  
AC CF 0006D 2\$:  
00EF 00072 3\$:  
007E 0007A  
007E 00082  
007E 0008A  
007E 00092  
0059 0009A  
007E 000A2  
007E 000AA  
007E 000B2  
007E 000BA

BRW  
CASEL  
.WORD

78-18,-  
198-18,-  
198-18,-  
198-18,-  
198-18,-  
198-18,-  
118-18  
198  
FROM DTYPE, #1, #36  
118-38,-

9039

50  
52

DC

30 04 AC D1 000BC  
2E 13 000C0  
00000000' EF 9F 000C2  
0085 31 000C8  
AC 0000014C 8F C5 000CB 4\$:  
50 63 A0 9E 000D4  
50 00000064 8F C7 000D8  
50 52 D0 000E0  
1F 50 D1 000E3  
03 15 000E6  
50 1F D0 000E8

CMPL  
BEQL  
PUSHAB  
BRW  
MULL3  
MOVAB  
DIVL3  
MOVL  
CMPL  
BLEQ  
MOVL  
FROM\_DTYPE, #48  
68  
P.AMN  
108  
#332, FROM\_LENGTH, RO  
99(RO), RO  
#100, RO, TO\_LENGTH  
TO\_LENGTH, RO  
RO, #31  
58  
#31, RO

9056  
9060  
9049  
9050

[illegible]

DBGEVALOP  
V04-000

C 16  
16-Sep-1984 00:32:25 VAX-11 Bliss-32 V4.0-742  
5-Sep-1984 21:54:24 [DEBUG.SRC]DBGEVALOP.B32;1

Page 272  
(40)

|    |    |    |    |       |       |       |       |                            |   |      |
|----|----|----|----|-------|-------|-------|-------|----------------------------|---|------|
|    |    | 52 | 2A | D0    | 00176 | 17\$: | MOVL  | #42, TO_LENGTH             | : | 9082 |
|    |    |    | 05 | 11    | 00179 |       | BRB   | 19\$                       | : |      |
| 52 | 0C | AC | 05 | C1    | 0017B | 18\$: | ADDL3 | #5, FROM_LENGTH, TO_LENGTH | : | 9084 |
|    |    | 50 | 52 | D0    | 00180 | 19\$: | MOVL  | TO_LENGTH, R0              | : | 9103 |
|    |    |    | 04 | 00183 |       |       | RET   |                            | : | 9104 |

; Routine Size: 388 bytes, Routine Base: DBG\$CODE + 1DD8

; 9011 9105 1

```
9013 9106 1 ROUTINE GET_SCALE(VAL_DESC, DIGITS) =
9014 9107 1
9015 9108 1 FUNCTION
9016 9109 1     This routine takes Floating-Point value descriptor data, convert its
9017 9110 1     value to text string, get the exponent and number of digits information
9018 9111 1     from the text string. Then we get the scaling factor from these
9019 9112 1     two numbers.
9020 9113 1
9021 9114 1 INPUT
9022 9115 1     VAL_DESC      - Floating-Point value descriptor.
9023 9116 1
9024 9117 1     DIGITS        - Address of the number of the digits.
9025 9118 1
9026 9119 1 OUTPUT
9027 9120 1     The scaling factor is returned.
9028 9121 1
9029 9122 1
9030 9123 2 BEGIN
9031 9124 2
9032 9125 2 MAP
9033 9126 2     DIGITS: REF VECTOR[1],      ! Address of the number of the
9034 9127 2                               ! digits.
9035 9128 2     VAL_DESC: REF DBG$VALDESC; ! Pointer to value descriptor.
9036 9129 2
9037 9130 2 LOCAL
9038 9131 2     BUFFER: VECTOR[50, BYTE],    ! Text string buffer
9039 9132 2     DESCRIPTOR: BLOCK[8, BYTE],  ! Vax standard descriptor.
9040 9133 2     DIGITS_IN_FRACT,             ! The number of digits in the fraction
9041 9134 2     D_PTR: REF VECTOR[.BYTE],    ! Pointer to text string at the '.'
9042 9135 2     E_FLAG,                     ! Flag set to indicate the E-format
9043 9136 2                               ! text string has 'E'
9044 9137 2     E_PTR: REF VECTOR[.BYTE],    ! Pointer to text string at E position
9045 9138 2     EXPONENT,                   ! The exponent of the floating-point
9046 9139 2     SCALE,                      ! The scaling factor
9047 9140 2     STATUS;                     ! Return status from library routine
9048 9141 2
9049 9142 2
9050 9143 2 ! Set up the descriptor.
9051 9144 2
9052 9145 2 DESCRIPTOR[DSC$B_CLASS] = DSC$K_CLASS_S;
9053 9146 2 DESCRIPTOR[DSC$B_DTYPE] = DSC$K_DTYPE_T;
9054 9147 2 DESCRIPTOR[DSC$W_LENGTH] = 50;
9055 9148 2 DESCRIPTOR[DSC$A_POINTER] = BUFFER;
9056 9149 2
9057 9150 2
9058 9151 2 ! Convert Floating-Point data to E formatted text string.
9059 9152 2
9060 9153 2 CASE .VAL_DESC[DBG$B_VALUE_DTYPE] FROM DSC$K_DTYPE_F TO DSC$K_DTYPE_H OF
9061 9154 2     SET
9062 9155 2     [DSC$K_DTYPE_F, DSC$K_DTYPE_D]:
9063 9156 2     BEGIN
9064 9157 2         CASE .VAL_DESC[DBG$B_VALUE_DTYPE] FROM DSC$K_DTYPE_F TO DSC$K_DTYPE_D OF
9065 9158 2             SET
9066 9159 2             [DSC$K_DTYPE_F]:
9067 9160 2                 DIGITS_IN_FRACT = 7;
9068 9161 2             [DSC$K_DTYPE_D]:
9069 9162 2                 DIGITS_IN_FRACT = 16;
```

```

9070      9163      TES;
9071      9164
9072      9165
9073      9166      ! In here, the exponent digits is default to 2.
9074      9167
9075      9168      STATUS = FOR$CVT D TE(.VAL_DESC[DBG$L_VALUE_POINTER], DESCRIPTOR,
9076      9169      .DIGITS_IN_FRACT);
9077      9170      END;
9078      9171
9079      9172      [DSC$K_DTYPE_G]:
9080      9173      BEGIN
9081      9174      DIGITS_IN_FRACT = 15;
9082      9175      STATUS = FOR$CVT G TE(.VAL_DESC[DBG$L_VALUE_POINTER], DESCRIPTOR,
9083      9176      .DIGITS_IN_FRACT, 0, 0, 3);
9084      9177      END;
9085      9178
9086      9179      [DSC$K_DTYPE_H]:
9087      9180      BEGIN
9088      9181      DIGITS_IN_FRACT = 33;
9089      9182      STATUS = FOR$CVT H TE(.VAL_DESC[DBG$L_VALUE_POINTER], DESCRIPTOR,
9090      9183      .DIGITS_IN_FRACT, 0, 0, 4);
9091      9184      END;
9092      9185
9093      9186      [INRANGE, OUTRANGE]:
9094      9187      $DBG_ERROR('DBGEVALOP\GET_SCALE, not floating-point data');
9095      9188
9096      9189      TES;
9097      9190
9098      9191      IF NOT .STATUS THEN $DBG_ERROR('DBGEVALOP\GET_SCALE, FOR$CVT_x_yE error');
9099      9192
9100      9193
9101      9194      ! Locate '+' or '-'.
9102      9195      ! (E+nn or E-nn for exp. <= 99), (+nnn or -nnn for exp. <=999)
9103      9196      ! is the standard Fortran E-format.
9104      9197
9105      9198      E_FLAG = CH$FIND_CH(50, BUFFER, %C'E');
9106      9199      E_PTR = CH$FIND_CH(50, BUFFER, %C'+');
9107      9200      IF .E_PTR EQL 0
9108      9201      THEN
9109      9202      E_PTR = CH$FIND_CH(50, BUFFER, %C'-');
9110      9203
9111      9204      IF .E_PTR EQL 0 THEN $DBG_ERROR('DBGEVALOP\GET_SCALE, not E-format expected');
9112      9205
9113      9206
9114      9207      ! Locate the '.'. For example: 0.48E+06.
9115      9208
9116      9209      D_PTR = CH$FIND_CH(50, BUFFER, %C'.');
9117      9210
9118      9211
9119      9212      ! Convert the next few characters into integer (+/- and digits).
9120      9213
9121      9214      DESCRIPTOR[DSC$W_LENGTH] = BUFFER[49] - .E_PTR + 1;
9122      9215      DESCRIPTOR[DSC$A_POINTER] = .E_PTR;
9123      9216      STATUS = OT$SCVT_TI_L(DESCRIPTOR, EXPONENT);
9124      9217      IF NOT .STATUS THEN $DBG_ERROR('DBGEVALOP\GET_SCALE, OT$SCVT_TI_L error');
9125      9218
9126      9219      DIGITS[0] = .E_PTR - .D_PTR - 1;
```

.PSECT DBG\$PLIT, NOWRT, SHR, PIC, 0

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |        |        |  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|--------|--------|--|
| 5F | 54 | 45 | 47 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 2C | 060D5 | P.AMP: | .ASCII | \,DBGEVALOP\<92>\GET_SCALE, not floating\  |
| 61 | 6F | 6C | 66 | 20 | 74 | 6F | 6E | 20 | 2C | 45 | 4C | 41 | 43 | 53 | 060E4 |        |        |  |
|    |    |    |    |    |    |    |    |    |    |    | 67 | 6E | 69 | 74 | 060F3 |        |        |  |
|    |    |    |    | 61 | 74 | 61 | 64 | 20 | 74 | 6E | 69 | 6F | 70 | 2D | 060F7 |        | .ASCII | \-point data\                              |
| 5F | 54 | 45 | 47 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 27 | 06102 | P.AMQ: | .ASCII | \'DBGEVALOP\<92>\GET_SCALE, FOR\$CVT_x_yE\ |
| 5F | 54 | 56 | 43 | 24 | 52 | 4F | 46 | 20 | 2C | 45 | 4C | 41 | 43 | 53 | 06111 |        |        |  |
|    |    |    |    |    |    |    |    |    |    |    | 45 | 79 | 5F | 78 | 06120 |        |        |  |
|    |    |    |    |    |    |    |    |    | 72 | 6F | 72 | 72 | 65 | 20 | 06124 |        | .ASCII | \ error\                                   |
| 5F | 54 | 45 | 47 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 2A | 0612A | P.AMR: | .ASCII | \*DBGEVALOP\<92>\GET_SCALE, not E-format\  |
| 6F | 66 | 2D | 45 | 20 | 74 | 6F | 6E | 20 | 2C | 45 | 4C | 41 | 43 | 53 | 06139 |        |        |  |
|    |    |    |    |    |    |    |    |    |    |    | 74 | 61 | 6D | 72 | 06148 |        |        |  |
|    |    |    |    |    |    | 64 | 65 | 74 | 63 | 65 | 70 | 78 | 65 | 20 | 0614C |        | .ASCII | \ expected\                                |
| 5F | 54 | 45 | 47 | 5C | 50 | 4F | 4C | 41 | 56 | 45 | 47 | 42 | 44 | 27 | 06155 | P.AMS: | .ASCII | \'DBGEVALOP\<92>\GET_SCALE, OT\$CVT_TI_L\  |
| 5F | 54 | 56 | 43 | 24 | 53 | 54 | 4F | 20 | 2C | 45 | 4C | 41 | 43 | 53 | 06164 |        |        |  |
|    |    |    |    |    |    |    |    |    |    |    | 4C | 5F | 49 | 54 | 06173 |        |        |  |
|    |    |    |    |    |    |    |    |    | 72 | 6F | 72 | 72 | 65 | 20 | 06177 |        | .ASCII | \ error\                                   |

.PSECT DBGS\$CODE,NOWRT, SHR, PIC.0

```
00FC 0000 GET_SCALE:
```

[illegible]

9106

9147  
9148  
9153

|           |      |          |      |    |    |       |      |        |                      |      |
|-----------|------|----------|------|----|----|-------|------|--------|----------------------|------|
|           |      |          |      | 57 | DD | 00050 | 28:  | PUSHL  | 88-18,-              |      |
|           |      |          |      | 01 | DD | 00052 |      | PUSHL  | 98-18                |      |
|           |      | 00028362 |      | 8F | DD | 00054 |      | PUSHL  | R7                   | 9187 |
|           | 66   |          |      | 03 | FB | 0005A |      | PUSHL  | #164706              |      |
|           |      |          |      | 53 | 11 | 0005D |      | CALLS  | #3, LIB\$SIGNAL      |      |
| 01        | 0A   | 16       |      | A2 | 8F | 0005F | 38:  | BRB    | 118                  |      |
|           | 0009 |          | 0004 |    |    | 00064 | 48:  | CASEB  | 22(R2), #10, #1      | 9157 |
|           |      |          |      |    |    |       |      | .WORD  | 58-48,-              |      |
|           | 50   |          |      | 07 | DD | 00068 | 58:  |        | 68-48                |      |
|           |      |          |      | 03 | 11 | 0006B |      | MOVL   | #7, DIGITS_IN_FRACT  | 9160 |
|           | 50   |          |      | 10 | DD | 0006D | 68:  | BRB    | 78                   |      |
|           |      |          |      | 50 | DD | 00070 | 78:  | MOVL   | #16, DIGITS_IN_FRACT | 9162 |
|           |      |          | 08   | AE | 9F | 00072 |      | PUSHL  | DIGITS_IN_FRACT      | 9169 |
|           |      |          | 18   | A2 | DD | 00075 |      | PUSHAB | DESCRIPTOR           | 9168 |
| 00000000G | 00   |          |      | 03 | FB | 00078 |      | PUSHL  | 24(R2)               |      |
|           |      |          |      | 2E | 11 | 0007F |      | CALLS  | #3, FOR\$CVT_D_TE    |      |
|           | 50   |          |      | 0F | DD | 00081 | 88:  | BRB    | 108                  |      |
|           |      |          |      | 03 | DD | 00084 |      | MOVL   | #15, DIGITS_IN_FRACT | 9174 |
|           |      |          |      | 7E | 7C | 00086 |      | PUSHL  | #3                   | 9175 |
|           |      |          |      | 50 | DD | 00088 |      | CLRQ   | -(SP)                |      |
|           |      |          | 14   | AE | 9F | 0008A |      | PUSHL  | DIGITS_IN_FRACT      | 9176 |
|           |      |          | 18   | A2 | DD | 0008D |      | PUSHAB | DESCRIPTOR           | 9175 |
| 00000000G | 00   |          |      | 06 | FB | 00090 |      | PUSHL  | 24(R2)               |      |
|           |      |          |      | 16 | 11 | 00097 |      | CALLS  | #6, FOR\$CVT_G_TE    |      |
|           | 50   |          |      | 21 | DD | 00099 | 98:  | BRB    | 108                  |      |
|           |      |          |      | 04 | DD | 0009C |      | MOVL   | #33, DIGITS_IN_FRACT | 9181 |
|           |      |          |      | 7E | 7C | 0009E |      | PUSHL  | #4                   | 9182 |
|           |      |          |      | 50 | DD | 000A0 |      | CLRQ   | -(SP)                |      |
|           |      |          | 14   | AE | 9F | 000A2 |      | PUSHL  | DIGITS_IN_FRACT      | 9183 |
|           |      |          | 18   | A2 | DD | 000A5 |      | PUSHAB | DESCRIPTOR           | 9182 |
| 00000000G | 00   |          |      | 06 | FB | 000A8 |      | PUSHL  | 24(R2)               |      |
|           | 53   |          |      | 50 | DD | 000AF | 108: | CALLS  | #6, FOR\$CVT_H_TE    |      |
|           | 0E   |          |      | 53 | E8 | 000B2 | 118: | MOVL   | R0, STATUS           |      |
|           |      |          | 2D   | A7 | 9F | 000B5 |      | BLBS   | STATUS, 128          | 9191 |
|           |      |          |      | 01 | DD | 000B8 |      | PUSHAB | P.AMQ                |      |
|           |      | 00028742 |      | 8F | DD | 000BA |      | PUSHL  | #1                   |      |
|           | 66   |          |      | 03 | FB | 000C0 |      | PUSHL  | #164706              |      |
| OC        | AE   | 32       |      | 8F | 3A | 000C3 | 128: | CALLS  | #3, LIB\$SIGNAL      |      |
|           |      |          |      | 02 | 12 | 000C9 |      | LOCC   | #69, #50, BUFFER     | 9198 |
|           |      |          |      | 51 | D4 | 000CB |      | BNEQ   | 138                  |      |
|           | 55   |          |      | 51 | DD | 000CD | 138: | CLRL   | R1                   |      |
| OC        | AE   | 32       |      | 2B | 3A | 000D0 |      | MOVL   | R1, E_FLAG           |      |
|           |      |          |      | 02 | 12 | 000D5 |      | LOCC   | #43, #50, BUFFER     | 9199 |
|           |      |          |      | 51 | D4 | 000D7 |      | BNEQ   | 148                  |      |
|           | 52   |          |      | 51 | DD | 000D9 | 148: | CLRL   | R1                   |      |
|           |      |          |      | 1C | 12 | 000DC |      | MOVL   | R1, E_PTR            |      |
| OC        | AE   | 32       |      | 2D | 3A | 000DE |      | BNEQ   | 168                  | 9200 |
|           |      |          |      | 02 | 12 | 000E3 |      | LOCC   | #45, #50, BUFFER     | 9202 |
|           |      |          |      | 51 | D4 | 000E5 |      | BNEQ   | 158                  |      |
|           | 52   |          |      | 51 | DD | 000E7 | 158: | CLRL   | R1                   |      |
|           |      |          |      | 0E | 12 | 000EA |      | MOVL   | R1, E_PTR            |      |
|           |      |          | 55   | A7 | 9F | 000EC |      | BNEQ   | 168                  | 9204 |
|           |      |          |      | 01 | DD | 000EF |      | PUSHAB | P.AMR                |      |
|           |      | 00028362 |      | 8F | DD | 000F1 |      | PUSHL  | #1                   |      |
|           | 66   |          |      | 03 | FB | 000F7 |      | PUSHL  | #164706              |      |
|           |      |          |      |    |    |       |      | CALLS  | #3, LIB\$SIGNAL      |      |

DBGEVALOP  
V04-000

H 16  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32:1

Page 277  
(41)

|    |           |    |    |    |       |       |        |                          |      |
|----|-----------|----|----|----|-------|-------|--------|--------------------------|------|
| 0C | AE        | 32 | 2E | 3A | 000FA | 16\$: | LOCC   | #46, #50, BUFFER         | 9209 |
|    |           |    | 02 | 12 | 000FF |       | BNEQ   | 17\$                     |      |
|    |           | 54 | 51 | D4 | 00101 |       | CLRL   | R1                       |      |
|    |           | 50 | 51 | D0 | 00103 | 17\$: | MOVL   | R1, D PTR                |      |
|    |           | 50 | AE | 9E | 00106 |       | MOVAB  | BUFFER+49, R0            | 9214 |
| 04 | AE        | 50 | 52 | C2 | 0010A |       | SUBL2  | E PTR, R0                |      |
|    |           | 50 | 01 | A1 | 0010D |       | ADDW3  | #T, R0, DESCRIPTOR       |      |
|    | 08        | AE | 52 | D0 | 00112 |       | MOVL   | E PTR, DESCRIPTOR+4      | 9215 |
|    |           |    | 5E | DD | 00116 |       | PUSHL  | SP                       | 9216 |
|    |           |    | AE | 9F | 00118 |       | PUSHAB | DESCRIPTOR               |      |
|    | 00000000G | 00 | C3 | FB | 0011B |       | CALLS  | #2, OTSSCVT_TI_L         |      |
|    |           | 53 | 50 | D0 | 00122 |       | MOVL   | R0, STATUS               |      |
|    |           | 0F | 53 | E8 | 00125 |       | BLBS   | STATUS, 18\$             | 9217 |
|    |           |    | C7 | 9F | 00128 |       | PUSHAB | P.AMS                    |      |
|    |           |    | 01 | DD | 0012C |       | PUSHL  | #1                       |      |
|    |           |    | 8F | DD | 0012E |       | PUSHL  | #164706                  |      |
|    |           | 66 | 03 | FB | 00134 |       | CALLS  | #3, LIB\$SIGNAL          |      |
| 51 |           | 52 | 54 | C3 | 00137 | 18\$: | SUBL3  | D PTR, E PTR, R1         | 9219 |
|    | 08        | BC | A1 | 9E | 0013B |       | MOVAB  | -T(R1), @DIGITS          |      |
|    |           |    | 55 | D5 | 00140 |       | TSTL   | E FLAG                   | 9220 |
|    |           |    | 03 | 13 | 00142 |       | BEQL   | 19\$                     |      |
|    |           |    | BC | D7 | 00144 |       | DECL   | @DIGITS                  |      |
| 50 |           | 6E | 08 | BC | C3    | 19\$: | SUBL3  | @DIGITS, EXPONENT, SCALE | 9221 |
|    |           |    | 08 | BC | 04    |       | RET    |                          | 9224 |
|    |           |    |    |    | 04    | 0014C |        |                          |      |

: Routine Size: 333 bytes, Routine Base: DBG\$CODE + 1F5C

: 9132 9225 1

```
9134 9226 1 ROUTINE FIND_JOIN (LEFT TYPE, RIGHT TYPE,  
9135 9227 1 NEW LEFT TYPE, NEW_RIGHT_TYPE, ROUT_INDEX,  
9136 9228 1 DEPTH, MIN,  
9137 9229 1 HIER_TBL, HIER_TBL_SIZE,  
9138 9230 1 INCOMP_TBL, INCOMP_TBL_SIZE,  
9139 9231 1 ROUT_TBL, ROUT_TBL_SIZE) =  
9140 9232 1  
9141 9233 1  
9142 9234 1  
9143 9235 1 FUNCTION  
9144 9236 1 Given a pair of types and a Type Hierarchy Graph, this routine attempts  
9145 9237 1 to determine what conversions should be done on the types. It does  
9146 9238 1 this by finding the "join" of the two nodes in the graph. Usually,  
9147 9239 1 this will just be a path from the lower type to the higher type; e.g.,  
9148 9240 1 if integer is added to float, float is the higher type, and a path  
9149 9241 1 from integer to float is found, so the integer is converted to float.  
9150 9242 1 There may be cases where both are converted to a higher type; e.g.,  
9151 9243 1 in FORTRAN, if FLOAT COMPLEX is added to D_FLOAT, both are promoted  
9152 9244 1 to D_FLOAT COMPLEX.  
9153 9245 1  
9154 9246 1 INPUTS  
9155 9247 1 HIER_TBL - Hier. Table  
9156 9248 1 HIER_TBL_SIZE - Hier. Table Size  
9157 9249 1 INCOMP_TBL - Incomp. Table  
9158 9250 1 INCOMP_TBL_SIZE - Incomp. Table Size  
9159 9251 1 ROUT_TBL - Rout. Table  
9160 9252 1 ROUT_TBL_SIZE - Rout. Table Size  
9161 9253 1 LEFT_TYPE - The type of the left operand.  
9162 9254 1 RIGHT_TYPE - The type of the right operand.  
9163 9255 1 NEW_LEFT_TYPE - The address in which to place the new type of the  
9164 9256 1 left operand.  
9165 9257 1 NEW_RIGHT_TYPE - The address in which to place the new type of the  
9166 9258 1 right operand.  
9167 9259 1 ROUT_INDEX - An address in which to fill in a routine index from  
9168 9260 1 the Operator Routine Table  
9169 9261 1 DEPTH - Depth of recursion. DBGSEVAL LANG_OPERATOR passes  
9170 9262 1 in zero, and this routine passes 1+.DEPTH when  
9171 9263 1 it calls itself, so it can keep track of the  
9172 9264 1 recursion depth.  
9173 9265 1 MIN - The minimum length conversion path so far.  
9174 9266 1  
9175 9267 1 OUTPUTS  
9176 9268 1 Routine Value: If a set of conversions to legal types was found,  
9177 9269 1 the length of the paths through the conversion graph is returned.  
9178 9270 1 Otherwise, MAX_DEPTH is returned. (I.e., a set of conversions were  
9179 9271 1 found if the return value is less than MAX_DEPTH. The reason for  
9180 9272 1 doing it this way (instead of just returning TRUE/FALSE) is so that  
9181 9273 1 a recursive method can be used to find the minimal path).  
9182 9274 1  
9183 9275 1 The following output parameters are filled in:  
9184 9276 1 NEW_LEFT_TYPE - The type to convert the left operand to.  
9185 9277 1 NEW_RIGHT_TYPE - The type to convert the right operand to.  
9186 9278 1 ROUT_INDEX - An index into the Operator Routine Table, pointing  
9187 9279 1 to a legal operator on the new types  
9188 9280 1  
9189 9281 2 BEGIN  
9190 9282 2
```

```

9191 9283 2
9192 9284 2
9193 9285 2
9194 9286 2
9195 9287 2
9196 9288 2
9197 9289 2
9198 9290 2
9199 9291 2
9200 9292 2
9201 9293 2
9202 9294 2
9203 9295 2
9204 9296 2
9205 9297 2
9206 9298 2
9207 9299 2
9208 9300 2
9209 9301 2
9210 9302 2
9211 9303 2
9212 9304 2
9213 9305 2
9214 9306 2
9215 9307 2
9216 9308 2
9217 9309 2
9218 9310 2
9219 9311 2
9220 9312 2
9221 9313 2
9222 9314 2
9223 9315 2
9224 9316 2
9225 9317 2
9226 9318 2
9227 9319 2
9228 9320 2
9229 9321 2
9230 9322 2
9231 9323 2
9232 9324 2
9233 9325 2
9234 9326 2
9235 9327 2
9236 9328 2
9237 9329 2
9238 9330 2
9239 9331 2
9240 9332 2
9241 9333 2
9242 9334 2
9243 9335 2
9244 9336 2
9245 9337 2
9246 9338 2
9247 9339 2

```

MAP

HIER\_TBL: REF VECTOR [WORD], ! Pointer to a Type Hierarchy Table  
INCOMP\_TBL: REF VECTOR [WORD], ! Pointer to a Type Incompatibility Table  
ROUT\_TBL: REF ORT\$TABLE; ! Pointer to an Operator Routine Table

LOCAL

HIER\_TBL\_ENTRY : TYPE\_GRAPH\$ENTRY, ! An entry in the Type Hierarchy Table.  
INCOMP\_TBL\_ENTRY: TYPE\_GRAPH\$ENTRY, ! An entry in the Type Incompatibility Table  
LEFT\_DONE: BYTE, ! Flag saying we have searched recursively from left sone  
LEFT\_FOUND: BYTE, ! A flag saying whether we found LEFT\_TYPE in the Type Hierarchy Table.  
LEFT\_INDEX, ! Index of the first occurrence of LEFT\_TYPE in the Type Hierarchy Table  
LEFT\_POTENTIAL\_TYPE, ! Potential new left type.  
POTENTIAL\_DEPTH, ! Possible minimum length path  
POTENTIAL\_INDEX, ! A candidate for an index into the Operator Routine Table  
REVERSE\_TYPES: TYPE\$PAIR, ! A type pair  
RIGHT\_DONE: BYTE, ! Flag saying we have searched recursively from the right son.  
RIGHT\_FOUND: BYTE, ! A flag saying whether we found RIGHT\_TYPE in the Type Hierarchy Table.  
RIGHT\_INDEX, ! Index of the first occurrence of RIGHT\_TYPE in the Type Hierarchy Table  
RIGHT\_POTENTIAL\_TYPE, ! Potential new right type  
TYPES: TYPE\$PAIR; ! A type pair.

! First see if the given type pair is legal, by searching the Operator Routine Table. If it is, then the length of the path to a legal type pair is zero and this is what we return.

TYPES [BS-LEFT TYPE] = .LEFT TYPE;  
TYPES [BS-RIGHT TYPE] = .RIGHT TYPE;  
INCR I FROM 0 TO .ROUT\_TBL SIZE - 1 DO  
IF .TYPES EQL .ROUT\_TBL [I, ORT\$W\_TYPES]  
THEN  
BEGIN  
.NEW\_LEFT TYPE = .LEFT TYPE;  
.NEW\_RIGHT TYPE = .RIGHT TYPE;  
.ROUT\_INDEX = .I;  
RETURN .DEPTH;  
END;

! Terminate further searching if minimum so far is already smaller than potential paths.

IF .DEPTH + 1 GEQ .MIN  
THEN

```
9248 9340 2      RETURN .MAX_DEPTH;
9249 9341 2
9250 9342 2      ! Check for incompatibility of the given types.
9251 9343 2      !
9252 9344 2      REVERSE_TYPES [BS_LEFT_TYPE] = .RIGHT_TYPE;
9253 9345 2      REVERSE_TYPES [BS_RIGHT_TYPE] = .LEFT_TYPE;
9254 9346 2      IF .INCOMP_TBL NEQ TABLEBASE
9255 9347 2      THEN
9256 9348 2          INCR I FROM 0 TO .INCOMP_TBL_SIZE - 1 DO
9257 9349 2              BEGIN
9258 9350 2                  INCOMP_TBL_ENTRY = .INCOMP_TBL [.I];
9259 9351 2                  IF .INCOMP_TBL_ENTRY EQL 0 THEN EXITLOOP;
9260 9352 2                  IF .TYPES EQL .INCOMP_TBL_ENTRY [TYPE_GRAPH$W_BOTH_TYPES]
9261 9353 2                  THEN
9262 9354 2                      RETURN .MAX_DEPTH;
9263 9355 2                  IF .REVERSE_TYPES EQL .INCOMP_TBL_ENTRY [TYPE_GRAPH$W_BOTH_TYPES]
9264 9356 2                  THEN
9265 9357 2                      RETURN .MAX_DEPTH;
9266 9358 2                  END;
9267 9359 2
9268 9360 2      ! Check for no Hierarchy Table being present.
9269 9361 2      !
9270 9362 2      IF .HIER_TBL EQL TABLEBASE
9271 9363 2      THEN
9272 9364 2          RETURN .MAX_DEPTH;
9273 9365 2
9274 9366 2      ! Find the first edges emanating from LEFT_TYPE and RIGHT_TYPE
9275 9367 2      ! in the Type Hierarchy Table.
9276 9368 2      !
9277 9369 2      LEFT_FOUND = FALSE;
9278 9370 2      RIGHT_FOUND = FALSE;
9279 9371 2      INCR I FROM 0 TO .HIER_TBL_SIZE - 1 DO
9280 9372 2          BEGIN
9281 9373 2              HIER_TBL_ENTRY = .HIER_TBL [.I];
9282 9374 2              IF .HIER_TBL_ENTRY EQL 0 THEN EXITLOOP;
9283 9375 2              IF NOT .LEFT_FOUND
9284 9376 2              THEN
9285 9377 2                  IF .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .LEFT_TYPE
9286 9378 2                  THEN
9287 9379 2                      BEGIN
9288 9380 2                          LEFT_FOUND = TRUE;
9289 9381 2                          LEFT_INDEX = .I;
9290 9382 2                          END;
9291 9383 2                  IF NOT .RIGHT_FOUND
9292 9384 2                  THEN
9293 9385 2                      IF .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .RIGHT_TYPE
9294 9386 2                      THEN
9295 9387 2                          BEGIN
9296 9388 2                              RIGHT_FOUND = TRUE;
9297 9389 2                              RIGHT_INDEX = .I;
9298 9390 2                              END;
9299 9391 2                          IF .LEFT_FOUND AND .RIGHT_FOUND THEN EXITLOOP;
9300 9392 2                      END;
9301 9393 2
9302 9394 2      ! Now do the recursive calls on the edges emanating out of the given
9303 9395 2      ! types. As a heuristic to speed up the search, start with the lower
9304 9396 2      ! index.
```

```
9305 9397 2 !
9306 9398 2 LEFT_DONE = FALSE;
9307 9399 2 RIGHT_DONE = FALSE;
9308 9400 2 WHILE TRUE DO
9309 9401 2 BEGIN
9310 9402 2 IF .LEFT_FOUND
9311 9403 2 THEN
9312 9404 2 BEGIN
9313 9405 2
9314 9406 2 ! Only do the recursive call from here if either we have already
9315 9407 2 ! searched from the right son, or if the left index comes before
9316 9408 2 ! the right index.
9317 9409 2
9318 9410 2 IF .RIGHT_DONE OR .LEFT_INDEX LEQ .RIGHT_INDEX
9319 9411 2 THEN
9320 9412 2 BEGIN
9321 9413 2 LEFT_DONE = TRUE;
9322 9414 2
9323 9415 2 ! Loop through all the successors to LEFT_TYPE.
9324 9416 2
9325 9417 2 HIER_TBL_ENTRY = .HIER_TBL [.LEFT_INDEX];
9326 9418 2 WHILE .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .LEFT_TYPE DO
9327 9419 2 BEGIN
9328 9420 2 POTENTIAL_DEPTH = FIND JOIN (
9329 9421 2 .HIER_TBL_ENTRY [TYPE_GRAPH$B_HIGHER_TYPE],
9330 9422 2 .RIGHT_TYPE,
9331 9423 2 LEFT_POTENTIAL_TYPE,
9332 9424 2 RIGHT_POTENTIAL_TYPE,
9333 9425 2 POTENTIAL_INDEX,
9334 9426 2 1+.DEPTH,
9335 9427 2 .MIN,
9336 9428 2 .HIER_TBL, .HIER_TBL_SIZE,
9337 9429 2 .INCOMP_TBL, .INCOMP_TBL_SIZE,
9338 9430 2 .ROUT_TBL, .ROUT_TBL_SIZE);
9339 9431 2
9340 9432 2 ! If we have a new minimum length path, then record
9341 9433 2 ! all the relevant information.
9342 9434 2
9343 9435 2 IF .POTENTIAL_DEPTH LSS .MIN
9344 9436 2 THEN
9345 9437 2 BEGIN
9346 9438 2 MIN = .POTENTIAL_DEPTH;
9347 9439 2 .NEW_LEFT_TYPE = .LEFT_POTENTIAL_TYPE;
9348 9440 2 .NEW_RIGHT_TYPE = .RIGHT_POTENTIAL_TYPE;
9349 9441 2 .ROUT_INDEX = .POTENTIAL_INDEX;
9350 9442 2 END;
9351 9443 2
9352 9444 2 ! Set up for next time around loop.
9353 9445 2
9354 9446 2 LEFT_INDEX = .LEFT_INDEX + 1;
9355 9447 2 HIER_TBL_ENTRY = .HIER_TBL [.LEFT_INDEX];
9356 9448 2 END;
9357 9449 2 END;
9358 9450 2
9359 9451 2 IF .RIGHT_FOUND
9360 9452 2 THEN
9361 9453 2
```

```
9362 9454 4
9363 9455 4
9364 9456 4
9365 9457 4
9366 9458 4
9367 9459 4
9368 9460 4
9369 9461 4
9370 9462 5
9371 9463 5
9372 9464 5
9373 9465 5
9374 9466 5
9375 9467 5
9376 9468 5
9377 9469 6
9378 9470 6
9379 9471 6
9380 9472 6
9381 9473 6
9382 9474 6
9383 9475 6
9384 9476 6
9385 9477 6
9386 9478 6
9387 9479 6
9388 9480 6
9389 9481 6
9390 9482 6
9391 9483 6
9392 9484 6
9393 9485 6
9394 9486 6
9395 9487 6
9396 9488 7
9397 9489 7
9398 9490 7
9399 9491 7
9400 9492 7
9401 9493 6
9402 9494 6
9403 9495 6
9404 9496 6
9405 9497 6
9406 9498 6
9407 9499 6
9408 9500 5
9409 9501 4
9410 9502 5
9411 9503 5
9412 9504 5
9413 9505 5
9414 9506 4
9415 9507 5
9416 9508 5
9417 9509 5
9418 9510 5
```

```
BEGIN
! Only do the recursive call from here if either we have already
! searched from the left son, or if the right index comes before
! the right index.
IF .LEFT_DONE OR .RIGHT_INDEX LEQ .LEFT_INDEX
THEN
  BEGIN
    RIGHT_DONE = TRUE;
    ! Loop through all the successors to RIGHT_TYPE.
    HIER_TBL_ENTRY = .HIER_TBL [.RIGHT_INDEX];
    WHILE .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .RIGHT_TYPE DO
      BEGIN
        POTENTIAL_DEPTH = FIND_JOIN (
          .LEFT_TYPE,
          .HIER_TBL_ENTRY [TYPE_GRAPH$B_HIGHER_TYPE],
          LEFT_POTENTIAL_TYPE,
          RIGHT_POTENTIAL_TYPE,
          POTENTIAL_INDEX,
          1+.DEPTH,
          .MIN,
          .HIER_TBL, .HIER_TBL_SIZE,
          .INCOMP_TBL, .INCOMP_TBL_SIZE,
          .ROUT_TBL, .ROUT_TBL_SIZE);
        ! If we have a new minimum length path, then record
        ! all the relevant information.
        IF .POTENTIAL_DEPTH LSS .MIN
        THEN
          BEGIN
            MIN = .POTENTIAL_DEPTH;
            .NEW_LEFT_TYPE = .LEFT_POTENTIAL_TYPE;
            .NEW_RIGHT_TYPE = .RIGHT_POTENTIAL_TYPE;
            .ROUT_INDEX = .POTENTIAL_INDEX;
          END;
        ! Set up for next time around loop.
        RIGHT_INDEX = .RIGHT_INDEX + 1;
        HIER_TBL_ENTRY = .HIER_TBL [.RIGHT_INDEX];
      END;
    END;
  END;
! If we have searched both paths, we are done.
IF (.RIGHT_DONE OR NOT .RIGHT_FOUND) AND (.LEFT_DONE OR NOT .LEFT_FOUND)
THEN
  EXITLOOP;
END;
```

9419 9511 2 RETURN .MIN;  
9420 9512 2  
9421 9513 1 END;

## OFFC 00000 FIND\_JOIN:

|    |    |    |          |    |       |        |                                      |      |
|----|----|----|----------|----|-------|--------|--------------------------------------|------|
| 52 | 08 | 5E | 0C       | C2 | 00002 | WORD   | Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 | 9226 |
|    |    | 08 | 04       | AC | F0    | SUBL2  | #12, SP                              |      |
|    |    | 5B | 08       | AC | D0    | INSV   | LEFT_TYPE, #8, #8, TYPES             | 9323 |
|    |    | 52 |          | 5B | 90    | MOVL   | RIGHT_TYPE, R11                      | 9324 |
|    |    | 50 |          | 01 | CE    | MOVB   | R11, TYPES                           |      |
|    |    |    |          | 1B | 11    | MNEGL  | #1, I                                | 9326 |
|    |    |    | 30       | BC | 40    | BRB    | 2\$                                  |      |
|    |    | 9E |          | 52 | B1    | PUSHAQ | @ROUT_TBL[I]                         |      |
|    |    |    |          | 12 | 12    | CMPW   | TYPES, @ (SP)+                       |      |
| 0C | BC |    | 04       | AC | D0    | BNEQ   | 2\$                                  |      |
| 10 | BC |    |          | 5B | D0    | MOVL   | LEFT_TYPE, @NEW_LEFT_TYPE            | 9329 |
| 14 | BC |    |          | 50 | D0    | MOVL   | R11, @NEW_RIGHT_TYPE                 | 9330 |
|    | 50 |    | 18       | AC | D0    | MOVL   | I, @ROUT_INDEX                       | 9331 |
|    |    |    |          |    | 04    | MOVL   | DEPTH, R0                            | 9332 |
|    |    |    |          |    |       | RET    |                                      |      |
| EO |    | 50 | 34       | AC | F2    | AOBLSS | ROUT_TBL_SIZE, I, 1\$                | 9326 |
| 56 |    | AC |          | 01 | C1    | ADDL3  | #1, DEPTH, R6                        | 9338 |
|    | 1C | AC |          | 56 | D1    | CMPL   | R6, MIN                              |      |
|    |    |    |          | 41 | 18    | BGEQ   | 6\$                                  |      |
| 53 | 08 | 08 |          | 5B | F0    | INSV   | R11, #8, #8, REVERSE TYPES           | 9344 |
|    |    | 53 | 04       | AC | 90    | MOVB   | LEFT_TYPE, REVERSE_TYPES             | 9345 |
|    |    | 50 | 00000000 | EF | 9E    | MOVAB  | TABLEBASE, R0                        | 9346 |
|    |    | 50 |          | 28 | AC    | CMPL   | INCOMP_TBL, R0                       |      |
|    |    |    |          | 1B | 13    | BEQL   | 5\$                                  |      |
|    |    | 50 |          | 01 | CE    | MNEGL  | #1, I                                | 9355 |
|    |    |    |          | 11 | 11    | BRB    | 4\$                                  |      |
|    |    | 51 | 28       | BC | 40    | MOVW   | @INCOMP_TBL[I], INCOMP_TBL_ENTRY     | 9350 |
|    |    |    |          | 0F | 13    | BEQL   | 5\$                                  | 9351 |
|    |    | 51 |          | 52 | B1    | CMPW   | TYPES, INCOMP_TBL_ENTRY              | 9352 |
|    |    |    |          | 1A | 13    | BEQL   | 6\$                                  |      |
|    |    | 51 |          | 53 | B1    | CMPW   | REVERSE_TYPES, INCOMP_TBL_ENTRY      | 9355 |
|    |    |    |          | 15 | 13    | BEQL   | 6\$                                  |      |
| EA |    | 50 | 2C       | AC | F2    | AOBLSS | INCOMP_TBL_SIZE, I, 3\$              | 9348 |
|    |    | 55 | 20       | AC | D0    | MOVL   | HIER_TBL, R5                         | 9362 |
|    |    | 50 | 00000000 | EF | 9E    | MOVAB  | TABLEBASE, R0                        |      |
|    |    | 50 |          | 55 | D1    | CMPL   | R5, R0                               |      |
|    |    |    |          | 08 | 12    | BNEQ   | 7\$                                  |      |
|    |    | 50 | 00000000 | EF | D0    | MOVL   | MAX_DEPTH, R0                        | 9364 |
|    |    |    |          |    | 04    | RET    |                                      |      |
|    |    |    |          | 59 | 94    | CLRB   | LEFT_FOUND                           | 9369 |
|    |    |    |          | 57 | 94    | CLRB   | RIGHT_FOUND                          | 9370 |
|    |    | 50 |          | 01 | CE    | MNEGL  | #1, I                                | 9371 |
|    |    |    |          | 2D | 11    | BRB    | 11\$                                 |      |
|    |    | 54 | 65       | 40 | B0    | MOVW   | (R5)[I], HIER_TBL_ENTRY              | 9373 |
|    |    |    |          | 2C | 13    | BEQL   | 12\$                                 | 9374 |
|    |    | 0E |          | 59 | E8    | BLBS   | LEFT_FOUND, 9\$                      | 9375 |
| 04 | AC |    |          | 00 | ED    | CMPZV  | #0, #8, HIER_TBL_ENTRY, LEFT_TYPE    | 9377 |
|    |    | 08 |          | 06 | 12    | BNEQ   | 9\$                                  |      |

|    |      |      |                |             |                                       |      |
|----|------|------|----------------|-------------|---------------------------------------|------|
| 59 | 01   | 90   | 000A5          | MOVB        | #1, LEFT FOUND                        | 9380 |
| 53 | 50   | D0   | 000A8          | MOVL        | I, LEFT INDEX                         | 9381 |
| 0D | 57   | E8   | 000AB          | 98: BLBS    | RIGHT FOUND, 10\$                     | 9383 |
| 08 | 00   | ED   | 000AE          | CMPZV       | #0, #8, HIER_TBL_ENTRY, R11           | 9385 |
|    | 06   | 12   | 000B3          | BNEQ        | 10\$                                  |      |
| 57 | 01   | 90   | 000B5          | MOVB        | #1, RIGHT FOUND                       | 9388 |
| 52 | 50   | D0   | 000B8          | MOVL        | I, RIGHT INDEX                        | 9389 |
| 03 | 59   | E9   | 000BB          | 10\$: BLBC  | LEFT FOUND, 11\$                      | 9391 |
| 05 | 57   | E8   | 000BE          | BLBS        | RIGHT FOUND, 12\$                     |      |
| 50 | 24   | AC   | F2 000C1       | 11\$: AOBLS | HIER_TBL_SIZE, I, 8\$                 | 9371 |
|    |      | 5A   | 94 000C6       | 12\$: CLRB  | LEFT_DONE                             | 9398 |
|    |      | 58   | 94 000C8       | CLRB        | RIGHT_DONE                            | 9399 |
| 5A | 59   | E9   | 000CA          | 13\$: BLBC  | LEFT FOUND, 18\$                      | 9452 |
| 05 | 58   | E8   | 000CD          | 14\$: BLBS  | RIGHT_DONE, 15\$                      | 9410 |
| 52 | 53   | D1   | 000D0          | CMPL        | LEFT_INDEX, RIGHT_INDEX               |      |
|    | 52   | 14   | 000D3          | BGTR        | 18\$                                  |      |
| 5A | 01   | 90   | 000D5          | 15\$: MOVB  | #1, LEFT DONE                         | 9413 |
| 54 | 6543 | B0   | 000D8          | 16\$: MOVW  | (R5)[LEFT_INDEX], HIER_TBL_ENTRY      | 9417 |
| 08 | 00   | ED   | 000DC          | CMPZV       | #0, #8, HIER_TBL_ENTRY, LEFT_TYPE     | 9418 |
|    | 43   | 12   | 000E2          | BNEQ        | 18\$                                  |      |
| 7E | 30   | AC   | 7D 000E4       | MOVQ        | ROUT_TBL, -(SP)                       | 9430 |
| 7E | 28   | AC   | 7D 000E8       | MOVQ        | INCOMP_TBL, -(SP)                     | 9429 |
|    | 24   | AC   | DD 000EC       | PUSHL       | HIER_TBL_SIZE                         | 9428 |
|    |      | 55   | DD 000EF       | PUSHL       | R5                                    |      |
|    | 1C   | AC   | DD 000F1       | PUSHL       | MIN                                   | 9427 |
|    |      | 56   | DD 000F4       | PUSHL       | R6                                    | 9426 |
|    | 20   | AE   | 9F 000F6       | PUSHAB      | POTENTIAL_INDEX                       | 9420 |
|    | 28   | AE   | 9F 000F9       | PUSHAB      | RIGHT_POTENTIAL_TYPE                  |      |
|    | 30   | AE   | 9F 000FC       | PUSHAB      | LEFT_POTENTIAL_TYPE                   |      |
|    |      | 5B   | DD 000FF       | PUSHL       | R11                                   | 9422 |
| 7E | 54   | 08   | EF 00101       | EXTZV       | #8, #8, HIER_TBL_ENTRY, -(SP)         | 9421 |
|    | FEF5 | CF   | 0D FB 00106    | CALLS       | #13, FIND_JOIN                        |      |
|    | 1C   | AC   | 50 D1 0010B    | CMPL        | POTENTIAL_DEPTH, MIN                  | 9435 |
|    |      | 12   | 18 0010F       | BGEQ        | 17\$                                  |      |
|    | 1C   | AC   | 50 D0 00111    | MOVL        | POTENTIAL_DEPTH, MIN                  | 9438 |
|    | 0C   | BC   | 08 AE D0 00115 | MOVL        | LEFT_POTENTIAL_TYPE, @NEW LEFT_TYPE   | 9439 |
|    | 10   | BC   | 04 AE D0 0011A | MOVL        | RIGHT_POTENTIAL_TYPE, @NEW RIGHT_TYPE | 9440 |
|    | 14   | BC   | 6E D0 0011F    | MOVL        | POTENTIAL_INDEX, @ROUT_INDEX          | 9441 |
|    |      | 53   | D6 00123       | 17\$: INCL  | LEFT_INDEX                            | 9446 |
|    |      | B1   | 11 00125       | BRB         | 16\$                                  | 9447 |
|    | 5A   | 57   | E9 00127       | 18\$: BLBC  | RIGHT FOUND, 22\$                     | 9452 |
|    | 05   | 5A   | E8 0012A       | BLBS        | LEFT_DONE, 19\$                       | 9460 |
|    | 53   | 52   | D1 0012D       | CMPL        | RIGHT_INDEX, LEFT_INDEX               |      |
|    |      | 52   | 14 00130       | BGTR        | 22\$                                  |      |
|    | 58   | 01   | 90 00132       | 19\$: MOVB  | #1, RIGHT DONE                        | 9463 |
|    | 54   | 6542 | B0 00135       | 20\$: MOVW  | (R5)[RIGHT_INDEX], HIER_TBL_ENTRY     | 9467 |
|    | 08   | 00   | ED 00139       | CMPZV       | #0, #8, HIER_TBL_ENTRY, R11           | 9468 |
|    |      | 44   | 12 0013E       | BNEQ        | 22\$                                  |      |
|    | 7E   | 30   | AC 7D 00140    | MOVQ        | ROUT_TBL, -(SP)                       | 9480 |
|    | 7E   | 28   | AC 7D 00144    | MOVQ        | INCOMP_TBL, -(SP)                     | 9479 |
|    |      | 24   | AC DD 00148    | PUSHL       | HIER_TBL_SIZE                         | 9478 |
|    |      | 55   | DD 0014B       | PUSHL       | R5                                    |      |
|    | 1C   | AC   | DD 0014D       | PUSHL       | MIN                                   | 9477 |
|    |      | 56   | DD 00150       | PUSHL       | R6                                    | 9476 |
|    | 20   | AE   | 9F 00152       | PUSHAB      | POTENTIAL_INDEX                       | 9470 |
|    | 28   | AE   | 9F 00155       | PUSHAB      | RIGHT_POTENTIAL_TYPE                  |      |
|    | 30   | AE   | 9F 00158       | PUSHAB      | LEFT_POTENTIAL_TYPE                   |      |

|    |      |    |    |      |       |       |                               |                                       |
|----|------|----|----|------|-------|-------|-------------------------------|---------------------------------------|
| 7E | 54   | 08 | 08 | EF   | 0015B | EXTZV | #8, #8, HIER_TBL_ENTRY, -(SP) | : 9472                                |
|    |      |    | 04 | AC   | DD    | 00160 | PUSHL                         | LEFT_TYPE                             |
|    | FE98 | CF |    | 0D   | FB    | 00163 | CALLS                         | #13, FIND_JOIN                        |
|    | 1C   | AC |    | 50   | D1    | 00168 | CMPL                          | POTENTIAL_DEPTH, MIN                  |
|    |      |    |    | 12   | 18    | 0016C | BGEQ                          | 21\$                                  |
|    | 1C   | AC |    | 50   | D0    | 0016E | MOVL                          | POTENTIAL_DEPTH, MIN                  |
|    | 0C   | BC | 08 | AE   | D0    | 00172 | MOVL                          | LEFT_POTENTIAL_TYPE, @NEW_LEFT_TYPE   |
|    | 10   | BC | 04 | AE   | D0    | 00177 | MOVL                          | RIGHT_POTENTIAL_TYPE, @NEW_RIGHT_TYPE |
|    | 14   | BC |    | 6E   | D0    | 0017C | MOVL                          | POTENTIAL_INDEX, @ROUT_INDEX          |
|    |      |    |    | 52   | D6    | 00180 | INCL                          | RIGHT_INDEX                           |
|    |      |    |    | B1   | 11    | 00182 | BRB                           | 20\$                                  |
|    | 06   |    |    | 58   | E8    | 00184 | BLBS                          | RIGHT_DONE, 23\$                      |
|    | 03   |    |    | 57   | E9    | 00187 | BLBC                          | RIGHT_FOUND, 23\$                     |
|    |      |    |    | FF3D | 31    | 0018A | BRW                           | 13\$                                  |
|    | 06   |    |    | 5A   | E8    | 0018D | BLBS                          | LEFT_DONE, 24\$                       |
|    | 03   |    |    | 59   | E9    | 00190 | BLBC                          | LEFT_FOUND, 24\$                      |
|    |      |    |    | FF37 | 31    | 00193 | BRW                           | 14\$                                  |
|    | 50   |    | 1C | AC   | D0    | 00196 | MOVL                          | MIN, R0                               |
|    |      |    |    | 04   | 0019A | RET   |                               | : 9511                                |
|    |      |    |    |      |       |       |                               | : 9513                                |

; Routine Size: 411 bytes, Routine Base: DBG\$CODE + 20A9

```
9423 9514 1 ROUTINE FIND_PATH (OLD_TYPE, NEW_TYPE, ROUT_INDEX,  
9424 9515 1     HIER_TBL, HIER_TBL_SIZE,  
9425 9516 1     INCOMP_TBL, INCOMP_TBL_SIZE,  
9426 9517 1     ROUT_TBL, ROUT_TBL_SIZE) =  
9427 9518 1  
9428 9519 1 FUNCTION  
9429 9520 1     This routine handles implicit type conversions on unary operators.  
9430 9521 1     It determines whether the given OLD_TYPE can be converted to  
9431 9522 1     a type that is legal for the current operator. The legal types  
9432 9523 1     can be determined using ORT_TBL. HIER_TBL gives the legal conversion  
9433 9524 1     paths. Thus, what this routine is doing is finding a path in  
9434 9525 1     HIER_TBL from the OLD_TYPE to a type that is legal for the  
9435 9526 1     current operator.  
9436 9527 1  
9437 9528 1     The value returned by the routine is the length of the shortest  
9438 9529 1     path to a legal type. If there are no paths, the value MAX_DEPTH  
9439 9530 1     is returned.  
9440 9531 1  
9441 9532 1     The search for a path uses the following recursive method:  
9442 9533 1     1) If the input type is legal, return 0  
9443 9534 1     2) If there are no edges out of the type, return MAX_DEPTH.  
9444 9535 1     3) For each edge out of the type, do a recursive call to obtain  
9445 9536 1     MIN(MAX_DEPTH, 1+FIND_PATH(edge)). Return the minimum of these.  
9446 9537 1  
9447 9538 1     *** Note - This routine may also require additional input to  
9448 9539 1     disambiguate cases where there are two or more paths of  
9449 9540 1     equal length.  
9450 9541 1  
9451 9542 1 INPUTS  
9452 9543 1     HIER_TBL - Hier. Table  
9453 9544 1     HIER_TBL_SIZE - Hier. Table Size  
9454 9545 1     INCOMP_TBL - Incomp. Table  
9455 9546 1     INCOMP_TBL_SIZE - Incomp. Table Size  
9456 9547 1     ROUT_TBL - Rout. Table  
9457 9548 1     ROUT_TBL_SIZE - Rout. Table Size  
9458 9549 1     OLD_TYPE - The type of the operand (A VAX standard type code.)  
9459 9550 1     NEW_TYPE - The address in which to leave the result type.  
9460 9551 1     ROUT_INDEX - The address in which to leave an index into the  
9461 9552 1     Operator Routine Table, pointing to a routine  
9462 9553 1     that handles the new type.  
9463 9554 1  
9464 9555 1 OUTPUTS  
9465 9556 1     NEW_TYPE - This output parameter is filled in with the type  
9466 9557 1     to convert to.  
9467 9558 1     ROUT_INDEX - Filled in with an index into the Operator Routine  
9468 9559 1     Table, pointing to a routine that handles the  
9469 9560 1     new type.  
9470 9561 1  
9471 9562 1 ROUTINE VALUE  
9472 9563 1     The length of the shortest path is returned. If this is equal to  
9473 9564 1     MAX_DEPTH then no path was found.  
9474 9565 1  
9475 9566 1  
9476 9567 2 BEGIN  
9477 9568 2  
9478 9569 2 MAP  
9479 9570 2     HIER_TBL: REF VECTOR [,WORD], ! Pointer to a Type Hierarchy Table
```

```
9480 9571 2      INCOMP_TBL: REF VECTOR [,WORD], : Pointer to a Type Incompatibility Table
9481 9572 2      ROUT_TBL: REF ORT$TABLE; : Pointer to an Operator Routine Table
9482 9573 2
9483 9574 2
9484 9575 2
9485 9576 2
9486 9577 2      LOCAL
9487 9578 2      FOUND, : A flag saying whether
9488 9579 2      HIER_TBL_ENTRY: TYPE_GRAPH$ENTRY, : we have found a new type
9489 9580 2      II, : An entry in the Type Hierarchy Table
9490 9581 2      MIN, : A loop counter
9491 9582 2      POTENTIAL_DEPTH, : The minimum depth so far
9492 9583 2      POTENTIAL_ROUT_INDEX, : A candidate for shortest depth
9493 9584 2      POTENTIAL_TYPE; : A candidate for routine index
9494 9585 2
9495 9586 2      : First see if the given type is legal, by searching the Operator
9496 9587 2      Routine Table. If it is, then the length of the path to a legal
9497 9588 2      type is zero and this is what we return.
9498 9589 2      INCR I FROM 0 TO .ROUT_TBL_SIZE - 1 DO
9499 9590 2      IF .OLD_TYPE EQL .ROUT_TBL [I, ORT$B_LEFT_TYPE]
9500 9591 2      THEN
9501 9592 2      BEGIN
9502 9593 2      .NEW_TYPE = .OLD_TYPE;
9503 9594 2      .ROUT_INDEX = .I;
9504 9595 2      RETURN 0;
9505 9596 2      END;
9506 9597 2
9507 9598 2      : Check for no Hierarchy Table being present.
9508 9599 2
9509 9600 2      IF .HIER_TBL EQL TABLEBASE
9510 9601 2      THEN
9511 9602 2      RETURN .MAX_DEPTH;
9512 9603 2
9513 9604 2      : Find the first edge emanating from OLD_TYPE in the Type Hierarchy Table.
9514 9605 2
9515 9606 2      FOUND = FALSE;
9516 9607 2      INCR I FROM 0 TO .HIER_TBL_SIZE - 1 DO
9517 9608 2      BEGIN
9518 9609 2      II = .I;
9519 9610 2      HIER_TBL_ENTRY = .HIER_TBL [I];
9520 9611 2      IF .HIER_TBL_ENTRY EQL 0 THEN EXITLOOP;
9521 9612 2      IF .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .OLD_TYPE
9522 9613 2      THEN
9523 9614 2      BEGIN
9524 9615 2      FOUND = TRUE;
9525 9616 2      EXITLOOP;
9526 9617 2      END;
9527 9618 2      END;
9528 9619 2
9529 9620 2      : If there were no edges emanating from OLD_TYPE, then no type conversion
9530 9621 2      can be done. Indicate this by returning MAX_DEPTH.
9531 9622 2
9532 9623 2      IF NOT .FOUND THEN RETURN .MAX_DEPTH;
9533 9624 2
9534 9625 2      : For each edge out of OLD_TYPE, chase down the target of that edge.
9535 9626 2      : Do a recursive call on that target to find a path from the target
9536 9627 2      : to a legal type. Keep track of the minimum length path in the
```

```
9537 9628 2 ! variable MIN.
9538 9629 2 !
9539 9630 2 MIN = .MAX_DEPTH;
9540 9631 2 WHILE .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .OLD_TYPE DO
9541 9632 2 BEGIN
9542 9633 2
9543 9634 2 POTENTIAL_DEPTH = FIND_PATH (
9544 9635 2 .HIER_TBL_ENTRY [TYPE_GRAPH$B_HIGHER_TYPE], POTENTIAL_TYPE,
9545 9636 2 POTENTIAL_ROUT_INDEX,
9546 9637 2 .HIER_TBL, .HIER_TBL_SIZE,
9547 9638 2 .INCOMP_TBL, .INCOMP_TBL_SIZE,
9548 9639 2 .ROUT_TBL, .ROUT_TBL_SIZE);
9549 9640 2
9550 9641 2 IF .POTENTIAL_DEPTH LSS .MIN
9551 9642 2 THEN
9552 9643 2 BEGIN
9553 9644 2 MIN = .POTENTIAL_DEPTH;
9554 9645 2 .NEW_TYPE = .POTENTIAL_TYPE;
9555 9646 2 .ROUT_INDEX = .POTENTIAL_ROUT_INDEX;
9556 9647 2 END;
9557 9648 2
9558 9649 2 ! Set up for next time around loop.
9559 9650 2 II = .II + 1;
9560 9651 2 HIER_TBL_ENTRY = .HIER_TBL [.II];
9561 9652 2 END;
9562 9653 2
9563 9654 2 IF .MIN EQL .MAX_DEPTH
9564 9655 2 THEN
9565 9656 2 RETURN .MAX_DEPTH
9566 9657 2 ELSE
9567 9658 2 RETURN .MIN + 1;
9568 9659 2
9569 9660 1 END; ! FIND_PATH
```

```
04 AC 9E 08 20 BC 40 7F 00011 1$:
08 BC 04 AC D0 0001D
0C BC 50 D0 00C22
E3 50 24 008D 31 00026
50 00000000' EF 9E 0002E 2$:
50 10 AC D1 00035
26 13 00039
51 D4 0003B
50 01 CE 0003D
17 11 00040
```

003C 00000 FIND\_PATH:

|        |                           |
|--------|---------------------------|
| .WORD  | Save R2,R3,R4,R5          |
| MOVAB  | MAX_DEPTH, R5             |
| SUBL2  | #8, SP                    |
| MNEGL  | #1, I                     |
| BRB    | 2\$                       |
| PUSHAQ | @ROUT_TBL[I]              |
| CMP2V  | #8, #8, @ (SP)+, OLD_TYPE |
| BNEQ   | 2\$                       |
| MOVL   | OLD_TYPE, @NEW_TYPE       |
| MOVL   | I, @ROUT_INDEX            |
| BRW    | 13\$                      |
| AOBLS  | ROUT_TBL_SIZE, I, 1\$     |
| MOVAB  | TABLEBASE, R0             |
| CMPL   | HIER_TBL, R0              |
| BEQL   | 6\$                       |
| CLRL   | FOUND                     |
| MNEGL  | #1, I                     |
| BRB    | 4\$                       |

```
9514
9590
9593
9594
9595
9590
9600
9606
9607
```

|    |    |      |    |    |      |    |       |       |        |                                   |  |      |
|----|----|------|----|----|------|----|-------|-------|--------|-----------------------------------|--|------|
|    |    |      | 53 |    | 50   | D0 | 00042 | 3\$:  | MOVL   | I, II                             |  | 9609 |
|    |    |      | 54 | 10 | BC40 | B0 | 00045 |       | MOVW   | @HIER_TBL[I], HIER_TBL_ENTRY      |  | 9610 |
|    |    |      |    |    | 12   | 13 | 0004A |       | BEQL   | 5\$                               |  | 9611 |
| 04 | AC | 54   | 08 |    | 00   | ED | 0004C |       | CMPZV  | #0, #8, HIER_TBL_ENTRY, OLD_TYPE  |  | 9612 |
|    |    |      |    |    | 05   | 12 | 00052 |       | BNEQ   | 4\$                               |  |      |
|    |    |      | 51 |    | 01   | D0 | 00054 |       | MOVL   | #1, FOUND                         |  | 9615 |
|    |    |      |    |    | 05   | 11 | 00057 |       | BRB    | 5\$                               |  | 9614 |
|    |    | E4   | 50 | 14 | AC   | F2 | 00059 | 4\$:  | AOBLSS | HIER_TBL_SIZE, I, 3\$             |  | 9607 |
|    |    |      | 04 |    | 51   | E8 | 0005E | 5\$:  | BLBS   | FOUND, 7\$                        |  | 9623 |
|    |    |      | 50 |    | 65   | D0 | 00061 | 6\$:  | MOVL   | MAX_DEPTH, R0                     |  |      |
|    |    |      |    |    | 04   |    | 00064 |       | RET    |                                   |  |      |
|    |    |      | 52 |    | 65   | D0 | 00065 | 7\$:  | MOVL   | MAX_DEPTH, MIN                    |  | 9630 |
| 04 | AC | 54   | 08 |    | 00   | ED | 00068 | 8\$:  | CMPZV  | #0, #8, HIER_TBL_ENTRY, OLD_TYPE  |  | 9631 |
|    |    |      |    |    | 36   | 12 | 0006E |       | BNEQ   | 10\$                              |  |      |
|    |    |      | 7E | 20 | AC   | 7D | 00070 |       | MOVQ   | ROUT_TBL, -(SP)                   |  | 9639 |
|    |    |      | 7E | 18 | AC   | 7D | 00074 |       | MOVQ   | INCOMP_TBL, -(SP)                 |  | 9638 |
|    |    |      | 7E | 10 | AC   | 7D | 00078 |       | MOVQ   | HIER_TBL, -(SP)                   |  | 9637 |
|    |    |      |    | 18 | AE   | 9F | 0007C |       | PUSHAB | POTENTIAL_ROUT_INDEX              |  | 9634 |
|    |    |      |    | 20 | AE   | 9F | 0007F |       | PUSHAB | POTENTIAL_TYPE                    |  |      |
| 7E |    | 54   | 08 |    | 08   | EF | 00082 |       | EXTZV  | #8, #8, HIER_TBL_ENTRY, -(SP)     |  | 9635 |
|    |    | FF74 | CF |    | 09   | FB | 00087 |       | CALLS  | #9, FIND_PATH                     |  |      |
|    |    |      | 52 |    | 50   | D1 | 0008C |       | CMPL   | POTENTIAL_DEPTH, MIN              |  | 9641 |
|    |    |      |    |    | 0C   | 18 | 0008F |       | BGEQ   | 9\$                               |  |      |
|    |    |      | 52 |    | 50   | D0 | 00091 |       | MOVL   | POTENTIAL_DEPTH, MIN              |  | 9644 |
|    | 08 |      | BC | 04 | AE   | D0 | 00094 |       | MOVL   | POTENTIAL_TYPE, @NEW_TYPE         |  | 9645 |
|    | 0C |      | BC |    | 6E   | D0 | 00099 |       | MOVL   | POTENTIAL_ROUT_INDEX, @ROUT_INDEX |  | 9646 |
|    |    |      |    |    | 53   | D6 | 0009D | 9\$:  | INCL   | II                                |  | 9650 |
|    |    |      | 54 | 10 | BC43 | B0 | 0009F |       | MOVW   | @HIER_TBL[II], HIER_TBL_ENTRY     |  | 9651 |
|    |    |      |    |    | C2   | 11 | 000A4 |       | BRB    | 8\$                               |  | 9631 |
|    |    |      | 65 |    | 52   | D1 | 000A6 | 10\$: | CMPL   | MIN, MAX_DEPTH                    |  | 9654 |
|    |    |      |    |    | 05   | 12 | 000A9 |       | BNEQ   | 11\$                              |  |      |
|    |    |      | 52 |    | 65   | D0 | 000AB |       | MOVL   | MAX_DEPTH, R2                     |  | 9658 |
|    |    |      |    |    | 02   | 11 | 000AE |       | BRB    | 12\$                              |  |      |
|    |    |      |    |    | 52   | D6 | 000B0 | 11\$: | INCL   | R2                                |  |      |
|    |    |      | 50 |    | 52   | D0 | 000B2 | 12\$: | MOVL   | R2, R0                            |  |      |
|    |    |      |    |    | 04   |    | 000B5 |       | RET    |                                   |  |      |
|    |    |      |    |    | 50   | D4 | 000B6 | 13\$: | CLRL   | R0                                |  | 9660 |
|    |    |      |    |    | 04   |    | 000B8 |       | RET    |                                   |  |      |

; Routine Size: 185 bytes. Routine Base: DBG\$CODE + 2244

```
9571 9661 1 ROUTINE FIND_PATH DEPOSIT (OLD_TYPE, NEW_TYPE, DEPTH,
9572 9662 1 HIER_TBL, HIER_TBL_SIZE,
9573 9663 1 INCOMP_TBL, INCOMP_TBL_SIZE,
9574 9664 1 ROUT_TBL, ROUT_TBL_SIZE) =
9575 9665 1
9576 9666 1 FUNCTION
9577 9667 1 This routine handles explicit type conversions on unary operators.
9578 9668 1 It is used to determine whether the type given in OLD_TYPE
9579 9669 1 can be converted to the type given in NEW_TYPE. The main
9580 9670 1 application of this is when the user is doing a DEPOSIT of a
9581 9671 1 value of type OLD_TYPE into a primary of type NEW_TYPE.
9582 9672 1 The routine may also get used to implement the "cast" operator
9583 9673 1 or to implement type conversions of subscript values.
9584 9674 1
9585 9675 1 This routine uses HIER_TBL and attempts to find a path from
9586 9676 1 OLD_TYPE to NEW_TYPE. The value TRUE is returned if a path
9587 9677 1 is found and FALSE is returned if a path is not found.
9588 9678 1
9589 9679 1 The search for a path uses the following recursive method:
9590 9680 1 1) If the two types are the same, return TRUE
9591 9681 1 2) If there are no edges out of OLD_TYPE, return FALSE.
9592 9682 1 3) For each edge out of OLD_TYPE, do a recursive call. If
9593 9683 1 any of these return TRUE, then return TRUE.
9594 9684 1
9595 9685 1 INPUTS
9596 9686 1 HIER_TBL - Hier. Table
9597 9687 1 HIER_TBL_SIZE - Hier. Table Size
9598 9688 1 INCOMP_TBL - Incomp. Table
9599 9689 1 INCOMP_TBL_SIZE - Incomp. Table Size
9600 9690 1 ROUT_TBL - Rout. Table
9601 9691 1 ROUT_TBL_SIZE - Rout. Table Size
9602 9692 1 OLD_TYPE - The type of the source operand (A VAX standard type code.)
9603 9693 1 NEW_TYPE - The target type (A VAX standard type code).
9604 9694 1 DEPTH - The current recursion depth
9605 9695 1
9606 9696 1 OUTPUTS
9607 9697 1 The routine value is one of:
9608 9698 1 TRUE - The source can be converted to the target type
9609 9699 1 FALSE - The source cannot be converted to the target type.
9610 9700 1
9611 9701 1 BEGIN
9612 9702 1
9613 9703 1 MAP
9614 9704 1 HIER_TBL: REF VECTOR [,WORD], ! Pointer to a Type Hierarchy Table
9615 9705 1 INCOMP_TBL: REF VECTOR [,WORD], ! Pointer to a Type Incompatibility Table
9616 9706 1 ROUT_TBL: REF ORTABLE; ! Pointer to an Operator Routine Table
9617 9707 1
9618 9708 1 LOCAL
9619 9709 1 FOUND, ! A flag saying whether
9620 9710 1 ! we have found a new type
9621 9711 1 HIER_TBL_ENTRY: TYPE_GRAPHSENTRY, ! An entry in the Type Hierarchy Table
9622 9712 1 II; ! A loop counter
9623 9713 1
9624 9714 1 ! First see if the source and target are already of the same type.
9625 9715 1
9626 9716 1
9627 9717 1
```

```
9628 9718 2
9629 9719
9630 9720
9631 9721
9632 9722
9633 9723
9634 9724
9635 9725
9636 9726
9637 9727
9638 9728
9639 9729
9640 9730
9641 9731
9642 9732
9643 9733
9644 9734
9645 9735
9646 9736
9647 9737
9648 9738
9649 9739
9650 9740
9651 9741
9652 9742
9653 9743
9654 9744
9655 9745
9656 9746
9657 9747
9658 9748
9659 9749
9660 9750
9661 9751
9662 9752
9663 9753
9664 9754
9665 9755
9666 9756
9667 9757
9668 9758
9669 9759
9670 9760
9671 9761
9672 9762
9673 9763
9674 9764
9675 9765
9676 9766
9677 9767
9678 9768
9679 9769
9680 9770
9681 9771
9682 9772
9683 9773
9684 9774

!
IF .NEW_TYPE EQL .OLD_TYPE
THEN
    RETURN TRUE;

! Then check whether we have already surpassed the maximum depth
! to which we have to search.
IF .DEPTH GTR .MAX_DEPTH
THEN
    RETURN FALSE;

! Check for no Hierarchy Table being present.
IF .HIER_TBL EQL TABLEBASE
THEN
    RETURN FALSE;

! Find the first edge emanating from OLD_TYPE in the Type Hierarchy Table.
FOUND = FALSE;
INCR I FROM 0 TO .HIER_TBL_SIZE - 1 DO
    BEGIN
        II = .I;
        HIER_TBL_ENTRY = .HIER_TBL [.I];
        IF .HIER_TBL_ENTRY EQL 0 THEN EXITLOOP;
        IF .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .OLD_TYPE
        THEN
            BEGIN
                FOUND = TRUE;
                EXITLOOP;
            END;
    END;

! If there were no edges emanating from OLD_TYPE, then no type conversion
! can be done. Indicate this by returning FALSE.
IF NOT .FOUND
THEN
    RETURN FALSE;

! For each edge out of OLD_TYPE, chase down the target of that edge.
! Do a recursive call on that target to find out whether there is
! a path from the edge.
WHILE .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .OLD_TYPE DO
    BEGIN
        IF FIND_PATH DEPOSIT (
            .HIER_TBL_ENTRY [TYPE_GRAPH$B_HIGHER_TYPE], .NEW_TYPE,
            .DEPTH+1,
            .HIER_TBL, .HIER_TBL_SIZE,
            .INCOMP_TBL, .INCOMP_TBL_SIZE,
            .ROUT_TBL, .ROUT_TBL_SIZE)
        THEN
            RETURN TRUE;
    END;
```

```

9685      9775      ! Set up for next time around loop.
9686      9776      II = .II + 1;
9687      9777      HIER_TBL_ENTRY = .HIER_TBL [.II];
9688      9778      END;
9689      9779
9690      9780      ! If we didn't return true from within the loop, then we failed to
9691      9781      ! find a path. Thus, return false here.
9692      9782      RETURN FALSE;
9693      9783
9694      9784      END; ! FIND_PATH_DEPOSIT
9695      9785
9696      9786
```

```

                                001C 00000 FIND_PATH_DEPOSIT:
                                .WORD Save R2,R3,R4
                                04 AC 08 AC D1 00002 CMPL NEW_TYPE, OLD_TYPE
                                00000000' EF 0C AC D1 00007 BEQL 5$
                                50 00000000' 6A 14 00011 CMPL DEPTH, MAX_DEPTH
                                50 10 AC D1 00013 BGTR 7$
                                50 10 AC D1 0001A MOVAB TABLEBASE, R0
                                50 10 AC D1 0001E CMPL HIER_TBL, R0
                                50 10 AC D1 0001E BEQL 7$
                                50 10 AC D1 00020 CLRL FOUND
                                50 10 AC D1 00022 MNEGL #1, 1
                                50 10 AC D1 00025 BRB 2$
                                52 10 AC D1 00027 1$: MOVL I, II
                                53 10 BC40 B0 0002A MOVW @HIER_TBL[I], HIER_TBL_ENTRY
                                12 13 0002F BEQL 3$
                                04 AC 53 08 00 ED 00031 CMPZV #0, #8, HIER_TBL_ENTRY, OLD_TYPE
                                51 01 D0 00037 BNEQ 2$
                                51 01 D0 00039 MOVL #1, FOUND
                                51 01 D0 0003C BRB 3$
                                E4 50 14 AC F2 0003E 2$: AOBLS HIER_TBL_SIZE, I, 1$
                                37 51 E9 00043 3$: BLBC FOUND, 7$
                                54 53 0C AC 01 C1 00046 ADDL3 #1, DEPTH, R4
                                53 08 00 ED 0004B 4$: CMPZV #0, #8, HIER_TBL_ENTRY, OLD_TYPE
                                7E 20 AC 7D 00051 BNEQ 7$
                                7E 18 AC 7D 00053 MOVQ ROUT_TBL, -(SP)
                                7E 10 AC 7D 00057 MOVQ INCOMP_TBL, -(SP)
                                54 DD 0005F MOVQ HIER_TBL, -(SP)
                                08 AC DD 00061 PUSHL R4
                                08 08 EF 00064 PUSHL NEW_TYPE
                                93 AF 09 FB 00069 EXTZV #8, #8, HIER_TBL_ENTRY, -(SP)
                                04 50 E9 0006D CALLS #9, FIND_PATH_DEPOSIT
                                50 01 D0 00070 5$: BLBC R0, 6$
                                04 00073 MOVL #1, R0
                                52 D6 00074 6$: RET
                                53 10 BC42 B0 00076 INCL II
                                CE 11 0007B MOVW @HIER_TBL[II], HIER_TBL_ENTRY
                                50 D4 0007D 7$: BRB 4$
                                04 0007F RET CLRL R0
                                9661
                                9719
                                9726
                                9732
                                9738
                                9739
                                9741
                                9742
                                9743
                                9744
                                9747
                                9746
                                9739
                                9755
                                9768
                                9763
                                9771
                                9770
                                9769
                                9768
                                9767
                                9773
                                9777
                                9778
                                9763
                                9786
```

DBGEVALOP  
V04-000

L 1  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32;1

Page 293  
(44)

; Routine Size: 128 bytes, Routine Base: DBG\$CODE + 22FD

; 9697 9787 1

```
9699 9788 1 ROUTINE FIXUP_EMPTY_SET(EMPTY_SET, VAL_DESC) =
9700 9789 1
9701 9790 1 FUNCTION
9702 9791 1 This routine sets up the typeid in empty set value descriptor
9703 9792 1 either by taking it from val_desc or creating a new one by
9704 9793 1 calling DBG$TYPEID_FOR_SET.
9705 9794 1
9706 9795 1 INPUTS
9707 9796 1 EMPTY_SET - [] value descriptor.
9708 9797 1
9709 9798 1 VAL_DESC - Value descriptor. This may be 0 for unary operation.
9710 9799 1 This may be another [], or standard value descriptor.
9711 9800 1
9712 9801 1 OUTPUTS
9713 9802 1 Fixed Empty Set Value Descriptor is returned.
9714 9803 1
9715 9804 1
9716 9805 2 BEGIN
9717 9806 2
9718 9807 2 MAP
9719 9808 2 EMPTY_SET: REF DBG$VALDESC,
9720 9809 2 VAL_DESC: REF DBG$VALDESC;
9721 9810 2
9722 9811 2
9723 9812 2 EMPTY_SET[DBG$B_DHDR_LANG] = .DBG$GB_LANGUAGE;
9724 9813 2 EMPTY_SET[DBG$B_VALUE_CLASS] = 0;
9725 9814 2 EMPTY_SET[DBG$B_VALUE_DTYPE] = 0;
9726 9815 2 IF .VAL_DESC EQL 0 OR
9727 9816 2 .VAL_DESC[DBG$B_DHDR_FCODE] NEQ RST$K_TYPE_SET OR
9728 9817 2 (.VAL_DESC[DBG$B_DHDR_LANG] EQL 'XX'FF' AND
9729 9818 2 .VAL_DESC[DBG$B_VALUE_CLASS] EQL 'XX'FF' AND
9730 9819 2 .VAL_DESC[DBG$B_VALUE_DTYPE] EQL 'XX'FF')
9731 9820 2 THEN
9732 9821 2 EMPTY_SET[DBG$L_DHDR_TYPEID] = DBG$TYPEID_FOR_SET(
9733 9822 2 DSC$K_DTYPE_L, RST$K_TYPE_SET, 256, TRUE)
9734 9823 2
9735 9824 2 ELSE
9736 9825 2 BEGIN
9737 9826 2 EMPTY_SET[DBG$L_DHDR_TYPEID] = .VAL_DESC[DBG$L_DHDR_TYPEID];
9738 9827 2 EMPTY_SET[DBG$W_VALUE_LENGTH] = .VAL_DESC[DBG$W_VALUE_LENGTH];
9739 9828 2 END;
9740 9829 2
9741 9830 2 RETURN .EMPTY_SET;
9742 9831 2 END;
```

## 000C 00000 FIXUP\_EMPTY\_SET:

|    |    |           |    |    |       |      |                         |      |
|----|----|-----------|----|----|-------|------|-------------------------|------|
|    | 53 | 04        | AC | D0 | 00002 | WORD | Save R2, R3             | 9788 |
| 03 | A3 | 00000000G | 00 | 90 | 00006 | MOVL | EMPTY_SET, R3           | 9812 |
|    |    | 16        | A3 | B4 | 0000E | MOVB | DBG\$GB_LANGUAGE, 3(R3) |      |
|    | 52 | 08        | AC | D0 | 00011 | CLRW | 22(R3)                  | 9814 |
|    |    |           | 1B | 13 | 00015 | MOVL | VAL_DESC, R2            | 9815 |
|    | 08 | 06        | A2 | 91 | 00017 | BEQL | 1\$                     |      |
|    |    |           |    |    |       | CMPL | 6(R2), #8               | 9816 |

DBGVALOP  
V04-000

N 1  
16-Sep-1984 00:32:25 VAX-11 Bliss-32 V4.0-742  
5-Sep-1984 21:54:24 [DEBUG.SRC]DBGVALOP.B32;1

Page 295  
(45)

|           |    |      |    |       |       |        |                         |   |      |
|-----------|----|------|----|-------|-------|--------|-------------------------|---|------|
|           |    |      | 15 | 12    | 0001B | BNEQ   | 1\$                     |   |      |
| FF        | 8F | 03   | A2 | 91    | 0001D | CMPS   | 3(R2), #255             | : | 9817 |
|           |    |      | 26 | 12    | 00022 | BNEQ   | 2\$                     | : |      |
| FF        | 8F | 17   | A2 | 91    | 00024 | CMPS   | 23(R2), #255            | : | 9818 |
|           |    |      | 1F | 12    | 00029 | BNEQ   | 2\$                     | : |      |
| FF        | 8F | 16   | A2 | 91    | 0002B | CMPS   | 22(R2), #255            | : | 9819 |
|           |    |      | 18 | 12    | 00030 | BNEQ   | 2\$                     | : |      |
|           |    |      | 01 | DD    | 00032 | PUSHL  | #1                      | : | 9821 |
|           | 7E | 0100 | 8F | 3C    | 00034 | MOVZWL | #256, -(SP)             | : |      |
|           |    |      | 08 | DD    | 00039 | PUSHL  | #8                      | : |      |
|           |    |      | 08 | DD    | 0003B | PUSHL  | #8                      | : |      |
| 00000000G | 00 |      | 04 | FB    | 0003D | CALLS  | #4, DBG\$TYPEID_FOR_SET | : |      |
| 08        | A3 |      | 50 | D0    | 00044 | MOVL   | R0, 8(R3)               | : |      |
|           |    |      | 0A | 11    | 00048 | BRB    | 3\$                     | : |      |
| 08        | A3 | 0    | A2 | D0    | 0004A | MOVL   | 8(R2), 8(R3)            | : | 9826 |
| 14        | A3 | 14   | A2 | B0    | 0004F | MOVW   | 20(R2), 20(R3)          | : | 9827 |
|           | 50 |      | 53 | D0    | 00054 | MOVL   | R3, R0                  | : | 9830 |
|           |    |      | 04 | 00057 | RET   |        |                         | : | 9831 |

; Routine Size: 88 bytes. Routine Base: DBG\$CODE + 237D

; 9743 9832 1

```

: 9745      9833 1 ROUTINE INTMED_DATA_FOR_DEP(FROM_DESC, TO_DATA_TYPE, FLAG) =
: 9746      9834 1
: 9747      9835 1 FUNCTION
: 9748      9836 1     There are times an intermediate data is needed before the deposit.
: 9749      9837 1     This routine takes the from descriptor converts to intermediate
: 9750      9838 1     data, return the intermediate data back to the caller. The caller
: 9751      9839 1     takes intermediate data then performs the deposit into TO_DESC.
: 9752      9840 1
: 9753      9841 1 INPUTS
: 9754      9842 1     FROM_DESC      - From value descriptor.
: 9755      9843 1
: 9756      9844 1     TO_DATA_TYPE   - Intermediate data type.
: 9757      9845 1
: 9758      9846 1     FLAG        - Flag set to TRUE to indidate left hand side
: 9759      9847 1     of the deposit.
: 9760      9848 1
: 9761      9849 1 OUTPUTS
: 9762      9850 1     Intermediate data type is returned.
: 9763      9851 1
: 9764      9852 1
: 9765      9853 2 BEGIN
: 9766      9854 2
: 9767      9855 2 MAP
: 9768      9856 2     FROM_DESC: REF DBG$VALDESC;      ! Pointer to value descriptor.
: 9769      9857 2
: 9770      9858 2 LOCAL
: 9771      9859 2     DIGITS,      ! The number of digits.
: 9772      9860 2     INTMED_DESC: REF DBG$VALDESC; ! Intermediate data value descriptor.
: 9773      9861 2
: 9774      9862 2
: 9775      9863 2 CASE .FROM_DESC[DBG$B_VALUE_DTYPE] FROM 0 TO DBG$K_MAXIMUM_DTYPE +
: 9776      9864 2     RST$K_TYPE_MAXIMUM OF
: 9777      9865 2 SET
: 9778      9866 2
: 9779      9867 2
: 9780      9868 2 [DSC$K_DTYPE_NL, DSC$K_DTYPE_NLO, DSC$K_DTYPE_NR, DSC$K_DTYPE_NRO,
: 9781      9869 2 DSC$K_DTYPE_NZ]:
: 9782      9870 2 BEGIN
: 9783      9871 2     IF .FLAG THEN RETURN .FROM_DESC;
: 9784      9872 2     INTMED_DESC = MAKE_VAL_DESC(.TO_DATA_TYPE,
: 9785      9873 2     DBG$NUM_BYTES(.TO_DATA_TYPE),
: 9786      9874 2     0,
: 9787      9875 2     TRUE);
: 9788      9876 2     INTMED_DESC[DBG$B_VALUE_DIGITS] = .FROM_DESC[DBG$B_VALUE_DIGITS];
: 9789      9877 2     INTMED_DESC[DBG$B_VALUE_SCALE] = .FROM_DESC[DBG$B_VALUE_SCALE];
: 9790      9878 2     INTMED_DESC[DBG$B_VALUE_LENGTH] = .FROM_DESC[DBG$B_VALUE_LENGTH];
: 9791      9879 2     INTMED_DESC = DBG$TYPE_CONV(.FROM_DESC, .INTMED_DESC);
: 9792      9880 2 END;
: 9793      9881 2
: 9794      9882 2
: 9795      9883 2 ! If right hand side is one of the floating-point data type, we need to
: 9796      9884 2 ! get the exponent value and that is the scaling factor. For
: 9797      9885 2 ! example F --> P, F = 0.1234567E+04 --> P = 1234.567. This should
: 9798      9886 2 ! not appear on the left hand side.
: 9799      9887 2
: 9800      9888 2 [DSC$K_DTYPE_F, DSC$K_DTYPE_D, DSC$K_DTYPE_G, DSC$K_DTYPE_H]:
: 9801      9889 2 BEGIN
```

```
9802      9890      3      INTMED_DESC = MAKE_VAL_DESC(.TO_DATA_TYPE,  
9803      9891      3      DBG$NUM_BYTES(.TO_DATA_TYPE),  
9804      9892      3      0,  
9805      9893      3      TRUE);  
9806      9894      3      INTMED_DESC[DBG$B_VALUE_SCALE] = GET_SCALE(.FROM_DESC, DIGITS);  
9807      9895      3      INTMED_DESC[DBG$B_VALUE_DIGITS] = .DIGITS;  
9808      9896      3      INTMED_DESC[DBG$W_VALUE_LENGTH] = .DIGITS;  
9809      9897      3      INTMED_DESC = DBG$TYPE_CONV (.FROM_DESC, .INTMED_DESC);  
9810      9898      3      END;  
9811      9899      3  
9812      9900      3  
9813      9901      3      ! If we deposit a interger data type into a scaled data type,  
9814      9902      3      change its class to be scaled. This can be on the left  
9815      9903      3      hand side of the deposit (may be volatile) or on the  
9816      9904      3      right hand side.  
9817      9905      3  
9818      9906      3      [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_L, DSC$K_DTYPE_O,  
9819      9907      3      DSC$K_DTYPE_BU, DSC$K_DTYPE_WU, DSC$K_DTYPE_LU, DSC$K_DTYPE_QU]:  
9820      9908      3      BEGIN  
9821      9909      3      INTMED_DESC = MAKE_VAL_DESC(.TO_DATA_TYPE,  
9822      9910      3      DBG$NUM_BYTES(.TO_DATA_TYPE),  
9823      9911      3      0,  
9824      9912      3      TRUE);  
9825      9913      3  
9826      9914      3  
9827      9915      3      ! Put in the scale information.  
9828      9916      3      !  
9829      9917      3      IF .FROM_DESC[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD  
9830      9918      3      THEN  
9831      9919      3      BEGIN  
9832      9920      3      INTMED_DESC[DBG$B_VALUE_SCALE] = .FROM_DESC[DBG$B_VALUE_SCALE];  
9833      9921      3  
9834      9922      3      ! The number of digits is assigned according to the COBOL rule.  
9835      9923      3      ! This should be adequate for all the languages.  
9836      9924      3      !  
9837      9925      3      !  
9838      9926      3      SELECT ONE .FROM_DESC[DBG$B_VALUE_DTYPE] OF  
9839      9927      3      SET  
9840      9928      3      [DSC$K_DTYPE_B, DSC$K_DTYPE_BU]:  
9841      9929      3      BEGIN  
9842      9930      3      IF .FROM_DESC[DBG$B_VALUE_DIGITS] EQL 0  
9843      9931      3      THEN  
9844      9932      3      BEGIN  
9845      9933      3      INTMED_DESC[DBG$W_VALUE_LENGTH] = 3;  
9846      9934      3      INTMED_DESC[DBG$B_VALUE_DIGITS] = 3;  
9847      9935      3      END  
9848      9936      3      ELSE  
9849      9937      3      BEGIN  
9850      9938      3      INTMED_DESC[DBG$W_VALUE_LENGTH] = .FROM_DESC[DBG$B_VALUE_DIGITS];  
9851      9939      3      INTMED_DESC[DBG$B_VALUE_DIGITS] = .FROM_DESC[DBG$B_VALUE_DIGITS];  
9852      9940      3      END;  
9853      9941      3      END;  
9854      9942      3      [DSC$K_DTYPE_W, DSC$K_DTYPE_WU]:  
9855      9943      3      BEGIN  
9856      9944      3      IF .FROM_DESC[DBG$B_VALUE_DIGITS] EQL 0  
9857      9945      3      THEN  
9858      9946      3
```

9859 9947 6  
9860 9948 6  
9861 9949 6  
9862 9950 6  
9863 9951 5  
9864 9952 6  
9865 9953 6  
9866 9954 6  
9867 9955 5  
9868 9956 5  
9869 9957 4  
9870 9958 4  
9871 9959 4  
9872 9960 5  
9873 9961 5  
9874 9962 5  
9875 9963 6  
9876 9964 6  
9877 9965 6  
9878 9966 6  
9879 9967 5  
9880 9968 6  
9881 9969 6  
9882 9970 6  
9883 9971 5  
9884 9972 5  
9885 9973 4  
9886 9974 4  
9887 9975 4  
9888 9976 5  
9889 9977 5  
9890 9978 5  
9891 9979 6  
9892 9980 6  
9893 9981 6  
9894 9982 6  
9895 9983 5  
9896 9984 6  
9897 9985 6  
9898 9986 6  
9899 9987 5  
9900 9988 5  
9901 9989 4  
9902 9990 4  
9903 9991 4  
9904 9992 4  
9905 9993 4  
9906 9994 4  
9907 9995 4  
9908 9996 4  
9909 9997 4  
9910 9998 4  
9911 9999 4  
9912 10000 4  
9913 10001 4  
9914 10002 4  
9915 10003 4

```
BEGIN
  INTMED_DESC[DBG$W_VALUE_LENGTH] = 5;
  INTMED_DESC[DBG$B_VALUE_DIGITS] = 5;
END
ELSE
  BEGIN
    INTMED_DESC[DBG$W_VALUE_LENGTH] = .FROM_DESC[DBG$B_VALUE_DIGITS];
    INTMED_DESC[DBG$B_VALUE_DIGITS] = .FROM_DESC[DBG$B_VALUE_DIGITS];
  END;
END;

[DSC$K_DTYPE_L, DSC$K_DTYPE_LU]:
BEGIN
  IF .FROM_DESC[DBG$B_VALUE_DIGITS] EQL 0
  THEN
    BEGIN
      INTMED_DESC[DBG$W_VALUE_LENGTH] = 10;
      INTMED_DESC[DBG$B_VALUE_DIGITS] = 10;
    END
  ELSE
    BEGIN
      INTMED_DESC[DBG$W_VALUE_LENGTH] = .FROM_DESC[DBG$B_VALUE_DIGITS];
      INTMED_DESC[DBG$B_VALUE_DIGITS] = .FROM_DESC[DBG$B_VALUE_DIGITS];
    END;
  END;

[DSC$K_DTYPE_Q, DSC$K_DTYPE_QU]:
BEGIN
  IF .FROM_DESC[DBG$B_VALUE_DIGITS] EQL 0
  THEN
    BEGIN
      INTMED_DESC[DBG$W_VALUE_LENGTH] = 20;
      INTMED_DESC[DBG$B_VALUE_DIGITS] = 20;
    END
  ELSE
    BEGIN
      INTMED_DESC[DBG$W_VALUE_LENGTH] = .FROM_DESC[DBG$B_VALUE_DIGITS];
      INTMED_DESC[DBG$B_VALUE_DIGITS] = .FROM_DESC[DBG$B_VALUE_DIGITS];
    END;
  END;
END;

TES;
END;

! There is no real conversion needed for left hand side
! of the deposit, we only want to make a place holder.
IF NOT .FLAG
THEN
  INTMED_DESC = DBG$TYPE_CONV(.FROM_DESC, .INTMED_DESC);
RETURN .INTMED_DESC;
```

9863



|      |    |      |    |       |          |              |                         |      |
|------|----|------|----|-------|----------|--------------|-------------------------|------|
| 06CE | C5 | 4008 | 8F | BB    | 000E0    | PUSHR        | #*M<R3, SP>             | 9894 |
| 1C   | A2 |      | 02 | FB    | 000E4    | CALLS        | #2, GET SCALE           |      |
| 1D   | A2 |      | 50 | 90    | 000E9    | MOVB         | R0, 28(INTMED_DESC)     | 9895 |
| 14   | A2 |      | 6E | 90    | 000ED    | MOVB         | DIGITS, 29(INTMED_DESC) | 9896 |
|      |    |      | 6E | B0    | 000F1    | MOVW         | DIGITS, 20(INTMED_DESC) | 9897 |
|      |    | 0098 | 31 | 000F5 | 6\$: BRW | 17\$         |                         | 9909 |
|      |    |      | 01 | DD    | 000FB    | 7\$: PUSHL   | #1                      |      |
|      |    |      | 7E | D4    | 000FA    | CLRL         | -(SP)                   |      |
|      |    | 08   | AC | DD    | 000FC    | PUSHL        | TO_DATA TYPE            | 9910 |
|      | 65 |      | 01 | FB    | 000FF    | CALLS        | #1, DBG\$NUM_BYTES      |      |
|      |    |      | 50 | DD    | 00102    | PUSHL        | R0                      |      |
|      |    | 08   | AC | DD    | 00104    | PUSHL        | TO_DATA TYPE            | 9909 |
|      | 66 |      | 04 | FB    | 00107    | CALLS        | #4, MAKE_VAL_DESC       |      |
|      | 52 |      | 50 | DD    | 0010A    | MOVL         | R0, INTMED_DESC         |      |
|      | 09 | 17   | A3 | 91    | 0010D    | CMPB         | 23(R3), #9              | 9917 |
|      |    |      | 79 | 12    | 00111    | BNEQ         | 16\$                    |      |
|      | 51 | 1C   | A2 | 9E    | 00113    | MOVAB        | 28(INTMED_DESC), R1     | 9920 |
|      | 50 | 1C   | A3 | 9E    | 00117    | MOVAB        | 28(R3), R0              |      |
|      | 61 |      | 60 | 90    | 0011B    | MOVB         | (R0), (R1)              |      |
|      | 02 |      | 54 | 91    | 0011E    | CMPB         | R4, #2                  | 9928 |
|      |    |      | 05 | 13    | 00121    | BEQL         | 8\$                     |      |
|      | 06 |      | 54 | 91    | 00123    | CMPB         | R4, #6                  |      |
|      |    |      | 0F | 12    | 00126    | BNEQ         | 9\$                     |      |
|      |    | 01   | A0 | 95    | 00128    | 8\$: TSTB    | 1(R0)                   | 9930 |
|      |    |      | 55 | 12    | 0012B    | BNEQ         | 15\$                    |      |
| 14   | A2 |      | 03 | B0    | 0012D    | MOVW         | #3, 20(INTMED_DESC)     | 9933 |
| 01   | A1 |      | 03 | 90    | 00131    | MOVB         | #3, 1(R1)               | 9934 |
|      |    |      | 55 | 11    | 00135    | BRB          | 16\$                    | 9930 |
|      | 03 |      | 54 | 91    | 00137    | 9\$: CMPB    | R4, #3                  | 9943 |
|      |    |      | 05 | 13    | 0013A    | BEQL         | 10\$                    |      |
|      | 07 |      | 54 | 91    | 0013C    | CMPB         | R4, #7                  |      |
|      |    |      | 0F | 12    | 0013F    | BNEQ         | 11\$                    |      |
|      |    | 01   | A0 | 95    | 00141    | 10\$: TSTB   | 1(R0)                   | 9945 |
|      |    |      | 3C | 12    | 00144    | BNEQ         | 15\$                    |      |
| 14   | A2 |      | 05 | B0    | 00146    | MOVW         | #5, 20(INTMED_DESC)     | 9948 |
| 01   | A1 |      | 05 | 90    | 0014A    | MOVB         | #5, 1(R1)               | 9949 |
|      |    |      | 3C | 11    | 0014E    | BRB          | 16\$                    | 9945 |
|      | 04 |      | 54 | 91    | 00150    | 11\$: CMPB   | R4, #4                  | 9959 |
|      |    |      | 05 | 13    | 00153    | BEQL         | 12\$                    |      |
|      | 08 |      | 54 | 91    | 00155    | CMPB         | R4, #8                  |      |
|      |    |      | 0F | 12    | 00158    | BNEQ         | 13\$                    |      |
|      |    | 01   | A0 | 95    | 0015A    | 12\$: TSTB   | 1(R0)                   | 9961 |
|      |    |      | 23 | 12    | 0015D    | BNEQ         | 15\$                    |      |
| 14   | A2 |      | 0A | B0    | 0015F    | MOVW         | #10, 20(INTMED_DESC)    | 9964 |
| 01   | A1 |      | 0A | 90    | 00163    | MOVB         | #10, 1(R1)              | 9965 |
|      |    |      | 23 | 11    | 00167    | BRB          | 16\$                    | 9961 |
|      | 05 |      | 54 | 91    | 00169    | 13\$: CMPB   | R4, #5                  | 9975 |
|      |    |      | 05 | 13    | 0016C    | BEQL         | 14\$                    |      |
|      | 09 |      | 54 | 91    | 0016E    | CMPB         | R4, #9                  |      |
|      |    |      | 19 | 12    | 00171    | BNEQ         | 16\$                    |      |
|      |    | 01   | A0 | 95    | 00173    | 14\$: TSTB   | 1(R0)                   | 9977 |
|      |    |      | 0A | 12    | 00176    | BNEQ         | 15\$                    |      |
| 14   | A2 |      | 14 | B0    | 00178    | MOVW         | #20, 20(INTMED_DESC)    | 9980 |
| 01   | A1 |      | 14 | 90    | 0017C    | MOVB         | #20, 1(R1)              | 9981 |
|      |    |      | 0A | 11    | 00180    | BRB          | 16\$                    | 9977 |
| 14   | A2 | 01   | A0 | 9B    | 00182    | 15\$: MOVZBW | 1(R0), 20(INTMED_DESC)  | 9985 |
| 01   | A1 | 01   | A0 | 90    | 00187    | MOVB         | 1(R0), 1(R1)            | 9986 |

```

M 2
16-Sep-1984 00:32:25      VAX-11 B115-32 V4.0-742
5-Sep-1984 21:54:24      [DEBUG.SRC]DBG EVALOP.B32;1

```

Page 302  
(46)

| PC   | OC | AC | EB | 0018C | 168: | BLBS  | FLAG, 188          |
|------|----|----|----|-------|------|-------|--------------------|
|      |    | 52 | DD | 00190 | 178: | PUSHL | INTMED_DESC        |
|      |    | 53 | DD | 00192 |      | PUSHL | R3                 |
| 042B | C5 | 02 | FB | 00194 |      | CALLS | #2, DBG\$TYPE CONV |
|      | 52 | 50 | DO | 00199 |      | MOVL  | RO, INTMED_DESC    |
|      | 50 | 52 | DO | 0019C | 188: | MOVL  | INTMED_DESC, RO    |
|      |    |    | 04 | 0019F |      | RET   |                    |

: 9928 10016 1

```
9930 10017 1
9931 10018 1
9932 10019 1
9933 10020 1
9934 10021 1
9935 10022 1
9936 10023 1
9937 10024 1
9938 10025 1
9939 10026 1
9940 10027 1
9941 10028 1
9942 10029 1
9943 10030 1
9944 10031 1
9945 10032 1
9946 10033 1
9947 10034 1
9948 10035 1
9949 10036 1
9950 10037 1
9951 10038 1
9952 10039 2
9953 10040 2
9954 10041 2
9955 10042 2
9956 10043 2
9957 10044 2
9958 10045 2
9959 10046 2
9960 10047 2
9961 10048 2
9962 10049 2
9963 10050 2
9964 10051 2
9965 10052 2
9966 10053 2
9967 10054 2
9968 10055 2
9969 10056 2
9970 10057 2
9971 10058 2
9972 10059 2
9973 10060 2
9974 10061 2
9975 10062 2
9976 10063 2
9977 10064 2
9978 10065 2
9979 10066 2
9980 10067 2
9981 10068 2
9982 10069 2
9983 10070 2
9984 10071 2
9985 10072 2
9986 10073 2
```

```
ROUTINE MAKE_VAL_DESC (TYPE, LENGTH, DESC, FLAG) =
FUNCTION
    Allocates space for a value descriptor of the given type, and
    fills in the fields.
INPUTS
    TYPE - A VAX standard dtype code.
    LENGTH - Length of the value descriptor
    DESC - Points to a Value descriptor which is
        used in some cases to determine the fields
        of this new descriptor. This is described
        in more detail in the code.
    FLAG - Flag set to indicate when call dbg$map_dtype_class
        the returned class should be SD instead of S for
        returning S case.
OUTPUTS
    Returns the address of a value descriptor allocated out of temporary
    memory.
BEGIN
MAP
    DESC: REF DBG$VALDESC;
LOCAL
    ALLOC_LENGTH,          ! Allocated length of descriptor
                             ! in bytes
    DESCR_LENGTH,          ! Length of the descriptor
    RESULT: REF DBG$VALDESC; ! Address of the result descriptor

DESCR_LENGTH = .LENGTH;
IF .TYPE EQL DSC$K_DTYPE_VT
THEN
    BEGIN
    IF .DESCR_LENGTH LSS 14
    THEN
        DESCR_LENGTH = 16
    ELSE
        DESCR_LENGTH = .DESCR_LENGTH + 2;
    RESULT = DBG$GET_TEMPMEM (ALLOC_LENGTH = DBG$K_VALDESC_BASE_SIZE +
        (3 + MIN(.DESCR_LENGTH, 256 + 2)) / 4);
    END
ELSE
    BEGIN
    IF .TYPE EQL DSC$K_DTYPE_P
    THEN
        DESCR_LENGTH = MAX (16, (.DESCR_LENGTH + 1) / 4)
    ELSE
        DESCR_LENGTH = MAX (16, .DESCR_LENGTH);
    RESULT = DBG$GET_TEMPMEM (ALLOC_LENGTH = DBG$K_VALDESC_BASE_SIZE +
        (3 + MIN(.DESCR_LENGTH, 256)) / 4);
    END;
```

```
9987 10074
9988 10075
9989 10076
9990 10077
9991 10078
9992 10079
9993 10080
9994 10081
9995 10082
9996 10083
9997 10084
9998 10085
9999 10086
10000 10087
10001 10088
10002 10089
10003 10090
10004 10091
10005 10092
10006 10093
10007 10094
10008 10095
10009 10096
10010 10097
10011 10098
10012 10099
10013 10100
10014 10101
10015 10102
10016 10103
10017 10104
10018 10105
10019 10106
10020 10107
10021 10108
10022 10109
10023 10110
10024 10111
10025 10112
10026 10113
10027 10114
10028 10115
```

```
! Fill in the fields of the new value descriptor.
!
RESULT [DBG$B_DHDR_LANG] = .DBG$B_LANGUAGE;
RESULT [DBG$B_DHDR_TYPE] = DBG$K_VALUE_DESC;
RESULT [DBG$W_DHDR_LENGTH] = 4*.XALLOC_LENGTH;
RESULT [DBG$B_DHDR_KIND] = RST$K_DATA;
IF .TYPE GTR DBG$K_MAXIMUM_DTYPE
THEN
  BEGIN
    RESULT [DBG$B_DHDR_FCODE] = .TYPE - DBG$K_MAXIMUM_DTYPE;
    RESULT [DBG$B_VALUE_CLASS] = 0;
    RESULT [DBG$B_VALUE_DTYPE] = 0;
  END
ELSE
  BEGIN
    IF .FLAG
    THEN
      RESULT [DBG$B_DHDR_FCODE] = RST$K_TYPE_DESCR
    ELSE
      RESULT [DBG$B_DHDR_FCODE] = RST$K_TYPE_ATOMIC;
    RESULT [DBG$B_VALUE_CLASS] = DBG$MAP_DTYPE_CLASS(.TYPE, .FLAG);
    RESULT [DBG$B_VALUE_DTYPE] = .TYPE;
  END;
RESULT [DBG$W_VALUE_LENGTH] = MIN(.LENGTH, 256);
RESULT [DBG$L_VALUE_POINTER] = RESULT[DBG$A_VALUE_ADDRESS];
! If we are converting to 'FIXED' then we need to fill in the
! dtype field. We also need to light the binscale flag.
!
IF .TYPE EQL DSC$K_DTYPE_FIXED
THEN
  BEGIN
    RESULT[DBG$V_VALUE_FL_BINSCALE] = 1;
    RESULT[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
  END;
RETURN .RESULT;
END; ! MAKE_VAL_DESC
```

```
001C 00000 MAKE_VAL_DESC:
50      08 AC D0 00002      .WORD      Save R2,R3,R4
54      04 AC D0 00006      MOVL        LENGTH, DESCR_LENGTH
25      54 D1 0000A      MOVL        TYPE, R4
      2A 12 0000D      CMPL        R4, #37
0E      50 D1 0000F      BNEQ        4$
      05 18 00012      CMPL        DESCR_LENGTH, #14
50      10 D0 00014      BGEQ        1$
      03 11 00017      MOVL        #16, DESCR_LENGTH
      BRB        2$
```

```
1001
1004
1005
1005
1005
```

|           |    |      |    |       |       |        |                             |                          |      |
|-----------|----|------|----|-------|-------|--------|-----------------------------|--------------------------|------|
|           | 50 | 02   | C0 | 00019 | 1\$:  | ADDL2  | #2, DESCR_LENGTH            | 1005                     |      |
|           | 51 | 50   | D0 | 0001C | 2\$:  | MOVL   | DESCR_LENGTH, R1            | 1005                     |      |
| 00000102  | 8F | 51   | D1 | 0001F |       | CMPL   | R1, #258                    |                          |      |
|           | 51 | 05   | 15 | 00026 |       | BLEQ   | 3\$                         |                          |      |
|           | 51 | 0102 | 8F | 3C    | 00028 | MOVZWL | #258, R1                    |                          |      |
|           | 51 | 03   | C0 | 0002D | 3\$:  | ADDL2  | #3, R1                      |                          |      |
|           | 51 | 04   | C6 | 00030 |       | DIVL2  | #4, R1                      |                          |      |
|           | 52 | 08   | A1 | 9E    | 00033 | MOVAB  | 8(R1), ALLOC_LENGTH         | 1005                     |      |
|           | 15 | 39   | 11 | 00037 |       | BRB    | 9\$                         |                          |      |
|           | 51 | 54   | D1 | 00039 | 4\$:  | CMPL   | R4, #21                     | 1006                     |      |
|           | 51 | 0E   | 12 | 0003C |       | BNEQ   | 5\$                         |                          |      |
|           | 51 | 01   | A0 | 9E    | 0003E | MOVAB  | 1(R0), R1                   | 1006                     |      |
|           | 51 | 04   | C6 | 00042 |       | DIVL2  | #4, R1                      |                          |      |
|           | 10 | 51   | D1 | 00045 |       | CMPL   | R1, #16                     |                          |      |
|           |    | 0A   | 19 | 00048 |       | BLSS   | 6\$                         |                          |      |
|           | 51 | 0B   | 11 | 0004A |       | BRB    | 7\$                         |                          |      |
|           | 10 | 50   | D0 | 0004C | 5\$:  | MOVL   | DESCR_LENGTH, R1            | 1006                     |      |
|           |    | 51   | D1 | 0004F |       | CMPL   | R1, #T6                     |                          |      |
|           | 51 | 03   | 18 | 00052 |       | BGEQ   | 7\$                         |                          |      |
|           | 50 | 10   | D0 | 00054 | 6\$:  | MOVL   | #16, R1                     |                          |      |
| 00000100  | 8F | 51   | D0 | 00057 | 7\$:  | MOVL   | R1, DESCR_LENGTH            |                          |      |
|           |    | 50   | D1 | 0005A |       | CMPL   | R0, #256                    | 1007                     |      |
|           | 50 | 05   | 15 | 00061 |       | BLEQ   | 8\$                         |                          |      |
|           | 50 | 0100 | 8F | 3C    | 00063 | MOVZWL | #256, R0                    |                          |      |
|           | 50 | 03   | C0 | 00068 | 8\$:  | ADDL2  | #3, R0                      |                          |      |
|           | 50 | 04   | C6 | 0006B |       | DIVL2  | #4, R0                      |                          |      |
|           | 52 | 08   | A0 | 9E    | 0006E | MOVAB  | 8(R0), ALLOC_LENGTH         | 1007                     |      |
|           |    | 52   | DD | 00072 | 9\$:  | PUSHL  | ALLOC_LENGTH                |                          |      |
| 00000000G | 00 | 01   | FB | 00074 |       | CALLS  | #1, DBG\$GET_TEMPMEM        |                          |      |
|           | 53 | 50   | D0 | 0007B |       | MOVL   | R0, RESULT                  |                          |      |
|           | 03 | 00   | 90 | 0007E |       | MOVB   | DBG\$GB_LANGUAGE, 3(RESULT) | 1007                     |      |
|           | 02 | 7A   | 8F | 90    | 00086 | MOVB   | #122, 2(RESULT)             | 1007                     |      |
| 63        | 52 |      | 04 | A5    | 0008B | MULW3  | #4, ALLOC_LENGTH, (RESULT)  | 1007                     |      |
|           | 07 |      | 06 | 90    | 0008F | MOVB   | #6, 7(RESULT)               | 1008                     |      |
|           | 52 | 14   | A3 | 9E    | 00093 | MOVAB  | 20(RESULT), R2              | 1008                     |      |
|           | 2B |      | 54 | D1    | 00097 | CMPL   | R4, #43                     | 1008                     |      |
|           |    |      | 0A | 15    | 0009A | BLEQ   | 10\$                        |                          |      |
| 06        | A3 |      | 2B | 83    | 0009C | SUBB3  | #43, R4, 6(RESULT)          | 1008                     |      |
|           |    | 02   | A2 | B4    | 000A1 | CLRW   | 2(R2)                       | 1008                     |      |
|           |    |      | 20 | 11    | 000A4 | BRB    | 13\$                        | 1008                     |      |
|           | 06 | 10   | AC | E9    | 000A6 | 10\$:  | BLBC                        | FLAG, 11\$               | 1009 |
|           | 06 |      | 03 | 90    | 000AA | MOVB   | #3, 6(RESULT)               | 1009                     |      |
|           | 06 |      | 04 | 11    | 000AE | BRB    | 12\$                        |                          |      |
|           |    |      | 02 | 90    | 000B0 | 11\$:  | MOVB                        | #2, 6(RESULT)            | 1009 |
|           |    | 10   | AC | DD    | 000B4 | 12\$:  | PUSHL                       | FLAG                     | 1009 |
|           |    |      | 54 | DD    | 000B7 |        | PUSHL                       | R4                       |      |
| F19F      | CF |      | 02 | FB    | 000B9 |        | CALLS                       | #2, DBG\$MAP_DTYPE_CLASS |      |
|           | 03 |      | 50 | 90    | 000BE |        | MOVB                        | R0, 3(R2)                |      |
|           | 02 |      | 54 | 90    | 000C2 |        | MOVB                        | R4, 2(R2)                | 1009 |
|           | 50 | 08   | AC | D0    | 000C6 | 13\$:  | MOVL                        | LENGTH, R0               | 1010 |
| 00000100  | 8F |      | 50 | D1    | 000CA |        | CMPL                        | R0, #256                 |      |
|           |    |      | 05 | 15    | 000D1 |        | BLEQ                        | 14\$                     |      |
|           | 50 | 0100 | 8F | 3C    | 000D3 | MOVZWL | #256, R0                    |                          |      |
|           | 62 |      | 50 | B0    | 000D8 | 14\$:  | MOVW                        | R0, (R2)                 |      |
|           | 18 | 20   | A3 | 9E    | 000DB | MOVAB  | 32(R3), 24(RESULT)          | 1010                     |      |
|           |    |      | 54 | D1    | 000E0 | CMPL   | R4, #43                     | 1010                     |      |
|           | 2B |      | 0B | 12    | 000E3 | BNEQ   | 15\$                        |                          |      |

DBGVALOP  
V04-000

L 2  
16-Sep-1984 00:32:25 VAX-11 B11ss-32 V4.0-742  
5-Sep-1984 21:54:24 [DEBUG.SRC]DBGVALOP.B32;1

Page 306  
(47)

|    |    |    |    |       |       |                |
|----|----|----|----|-------|-------|----------------|
| 1E | A3 | 08 | 88 | 000E5 | BISB2 | #8, 30(RESULT) |
| 02 | A2 | 08 | 90 | 000E9 | MOVB  | #8, 2(R2)      |
|    | 50 | 53 | D0 | 000ED | MOVL  | RESULT, R0     |
|    |    |    | 04 | 000F0 | RET   |                |

: 1011  
: 1011  
: 1011  
: 1011

; Routine Size: 241 bytes, Routine Base: DBG\$CODE + 2575

```
10030 10116 1
10031 10117 1
10032 10118 1
10033 10119 1
10034 10120 1
10035 10121 1
10036 10122 1
10037 10123 1
10038 10124 1
10039 10125 1
10040 10126 1
10041 10127 1
10042 10128 1
10043 10129 1
10044 10130 1
10045 10131 1
10046 10132 1
10047 10133 1
10048 10134 1
10049 10135 1
10050 10136 1
10051 10137 1
10052 10138 1
10053 10139 1
10054 10140 1
10055 10141 1
10056 10142 1
10057 10143 1
10058 10144 1
10059 10145 1
10060 10146 1
10061 10147 1
10062 10148 1
10063 10149 1
10064 10150 1
10065 10151 1
10066 10152 1
10067 10153 1
10068 10154 1
10069 10155 1
10070 10156 1
10071 10157 1
10072 10158 1
10073 10159 1
10074 10160 1
10075 10161 1
10076 10162 1
10077 10163 1
10078 10164 1
10079 10165 1
10080 10166 1
10081 10167 1
10082 10168 1
10083 10169 1
10084 10170 1
10085 10171 1
10086 10172 1
```

ROUTINE MAP\_NRO\_DTYPE\_IN\_RPG(VAL\_DESC) =

## FUNCTION

This routine maps the representation of least significant digit and sign into RPG standard output format accepting both normal or alternate as inputs.

## INPUTS

VAL\_DESC - Pointer to NRO data type value descriptor.

## OUTPUTS

VAL\_DESC is returned.

## BEGIN

## MAP

VAL\_DESC: REF DBG\$VALDESC; ! Pointer to value descriptor

## LOCAL

LEAST\_SIG\_DIGIT: REF VECTOR[.BYTE]; ! Pointer to the least significant digit

! In RPG, representation of least significant digit and sign both normal/alternate ASCII char. can be accepted in RPG. but RPG choose one format to be the standard output format. In here we simply map it into the standard output format.

LEAST\_SIG\_DIGIT = .VAL\_DESC[DBG\$VALUE\_POINTER] +  
.VAL\_DESC[DBG\$VALUE\_LENGTH] - 1;

We perform the following mapping:

| Input | alt.  | Output |
|-------|-------|--------|
| norm. |       |        |
| {     | 0 [ ? | 0      |
| A     | 1     | 1      |
| B     | 2     | 2      |
| C     | 3     | 3      |
| D     | 4     | 4      |
| E     | 5     | 5      |
| F     | 6     | 6      |
| G     | 7     | 7      |
| H     | 8     | 8      |
| I     | 9     | 9      |
| }     | ! :   | }      |

IF .LEAST\_SIG\_DIGIT[0] EQL '(' OR  
.LEAST\_SIG\_DIGIT[0] EQL '0' OR  
.LEAST\_SIG\_DIGIT[0] EQL '[' OR  
.LEAST\_SIG\_DIGIT[0] EQL '?'

## THEN

LEAST\_SIG\_DIGIT[0] = '0';

IF .LEAST\_SIG\_DIGIT[0] GEQ 'A' AND

```
:10087      10173  2
:10088      10174  2
:10089      10175  2
:10090      10176  2
:10091      10177  2
:10092      10178  2
:10093      10179  2
:10094      10180  2
:10095      10181  2
:10096      10182  2
:10097      10183  2
:10098      10184  2
:10099      10185  1
```

```
      .LEAST_SIG_DIGIT[0] LEQ '1'
THEN
      LEAST_SIG_DIGIT[0] = .LEAST_SIG_DIGIT[0] - %X'10';

IF .LEAST_SIG_DIGIT[0] EQL '}' OR
   .LEAST_SIG_DIGIT[0] EQL ']' OR
   .LEAST_SIG_DIGIT[0] EQL ':' OR
   .LEAST_SIG_DIGIT[0] EQL '!'
THEN
      LEAST_SIG_DIGIT[0] = '}';

RETURN .VAL_DESC;
END;
```

| 0000 00000 MAP_NRO_DTYPE_IN RPG: |    |    |    |       |       |        |                          |  |      |
|----------------------------------|----|----|----|-------|-------|--------|--------------------------|--|------|
|                                  | 51 | 04 | AC | D0    | 00002 | .WORD  | Save nothing             |  | 1011 |
|                                  | 50 | 14 | A1 | 3C    | 00006 | MOVL   | VAL_DESC, R1             |  | 1014 |
|                                  | 50 | 18 | A1 | C0    | 0000A | MOVZWL | 20(R1), R0               |  | 1014 |
| 7B                               | 8F |    | 70 | 91    | 0000E | ADDL2  | 24(R1), R0               |  |      |
|                                  |    |    | 10 | 13    | 00012 | CMPB   | -(LEAST_SIG_DIGIT), #123 |  | 1016 |
|                                  | 30 |    | 60 | 91    | 00014 | BEQL   | 1\$                      |  |      |
|                                  |    |    | 0B | 13    | 00017 | CMPB   | (LEAST_SIG_DIGIT), #48   |  | 1016 |
| 5B                               | 8F |    | 60 | 91    | 00019 | BEQL   | 1\$                      |  |      |
|                                  |    |    | 05 | 13    | 0001D | CMPB   | (LEAST_SIG_DIGIT), #91   |  | 1016 |
|                                  | 3F |    | 60 | 91    | 0001F | BEQL   | 1\$                      |  |      |
|                                  |    |    | 03 | 12    | 00022 | CMPB   | (LEAST_SIG_DIGIT), #63   |  | 1016 |
|                                  | 60 |    | 30 | 90    | 00024 | BNEQ   | 2\$                      |  |      |
| 41                               | 8F |    | 60 | 91    | 00027 | MOVB   | #48, (LEAST_SIG_DIGIT)   |  | 1017 |
|                                  |    |    | 09 | 1F    | 0002B | CMPB   | (LEAST_SIG_DIGIT), #65   |  | 1017 |
| 49                               | 8F |    | 60 | 91    | 0002D | BLSSU  | 3\$                      |  |      |
|                                  |    |    | 03 | 1A    | 00031 | CMPB   | (LEAST_SIG_DIGIT), #73   |  | 1017 |
|                                  | 60 |    | 10 | 82    | 00033 | BGTRU  | 3\$                      |  |      |
| 7D                               | 8F |    | 60 | 91    | 00036 | SUBB2  | #16, (LEAST_SIG_DIGIT)   |  | 1017 |
|                                  |    |    | 10 | 13    | 0003A | CMPB   | (LEAST_SIG_DIGIT), #125  |  | 1017 |
| 5D                               | 8F |    | 60 | 91    | 0003C | BEQL   | 4\$                      |  |      |
|                                  |    |    | 0A | 13    | 00040 | CMPB   | (LEAST_SIG_DIGIT), #93   |  | 1017 |
|                                  | 3A |    | 60 | 91    | 00042 | BEQL   | 4\$                      |  |      |
|                                  |    |    | 05 | 13    | 00045 | CMPB   | (LEAST_SIG_DIGIT), #58   |  | 1017 |
|                                  | 21 |    | 60 | 91    | 00047 | BEQL   | 4\$                      |  |      |
|                                  |    |    | 04 | 12    | 0004A | CMPB   | (LEAST_SIG_DIGIT), #33   |  | 1018 |
|                                  | 60 | 7D | 8F | 90    | 0004C | BNEQ   | 5\$                      |  |      |
|                                  | 50 |    | 51 | D0    | 00050 | MOVB   | #125, (LEAST_SIG_DIGIT)  |  | 1018 |
|                                  |    |    | 04 | 00053 | MOVL  | R1, R0 |                          |  | 1018 |
|                                  |    |    |    |       | RET   |        |                          |  | 1018 |

; Routine Size: 84 bytes,      Routine Base: DBG\$CODE + 2666

:10100            10186 1

```
:10102      10187  1
:10103      10188  1
:10104      10189  1
:10105      10190  1
:10106      10191  1
:10107      10192  1
:10108      10193  1
:10109      10194  1
:10110      10195  1
:10111      10196  1
:10112      10197  1
:10113      10198  1
:10114      10199  1
:10115      10200  1
:10116      10201  1
:10117      10202  1
:10118      10203  1
:10119      10204  1
:10120      10205  1
:10121      10206  1
:10122      10207  1
:10123      10208  1
:10124      10209  1
:10125      10210  2
:10126      10211  2
:10127      10212  2
:10128      10213  2
:10129      10214  2
:10130      10215  2
:10131      10216  2
:10132      10217  2
:10133      10218  2
:10134      10219  2
:10135      10220  2
:10136      10221  2
:10137      10222  2
:10138      10223  2
:10139      10224  2
:10140      10225  2
:10141      10226  2
:10142      10227  2
:10143      10228  2
:10144      10229  2
:10145      10230  2
:10146      10231  2
:10147      10232  2
:10148      10233  2
:10149      10234  2
:10150      10235  2
:10151      10236  2
:10152      10237  2
:10153      10238  2
:10154      10239  2
:10155      10240  2
:10156      10241  2
:10157      10242  2
:10158      10243  2
```

ROUTINE MAP\_PACKED (NEW\_TYPE, DIGITS, OTHER\_TYPE): NOVALUE =

## FUNCTION

This routine maps a packed decimal type to another type. This cannot always be done in the hierarchy tables; in the case of converting packed to floating point, it is necessary to know how many digits are involved before mapping the type.

## INPUTS

NEW\_TYPE - The address in which to place the new type. On entering the routine, it should already point to type DSCSK\_DTYPE\_P.

DIGITS - The number of digits in the packed decimal number.

OTHER\_TYPE - In certain instances, the other operand helps in determining how to map the packed decimal number. For example, if there are 15 digits in the packed decimal number: it is mapped to gfloat if the other operand is gfloat, hfloat if the other operand is hfloat, otherwise double.

## OUTPUTS

A new type may be returned in NEW\_TYPE.

## BEGIN

! Expect NEW\_TYPE to already point to a packed decimal type. This assures that we have at least one packed decimal operand, and can case on the other.

IF ..NEW\_TYPE NEQ DSCSK\_DTYPE\_P

THEN

RETURN;

CASE .OTHER\_TYPE FROM 0 TO DBG\$K\_MAXIMUM\_DTYPE + RST\$K\_TYPE\_MAXIMUM OF  
SET

! If the other operand is floating, then the conversion is packed -> float. The type of floating point conversion (whether to single or double precision, or to gfloat or hfloat) depends on both the number of digits in the packed decimal number, and in some cases, on the type of the other operand.

[DSCSK\_DTYPE\_F, DSCSK\_DTYPE\_D, DSCSK\_DTYPE\_G, DSCSK\_DTYPE\_H]:

BEGIN

CASE .DIGITS FROM 1 TO DBG\$K\_LARGEST\_PACKED OF  
SET

[1 TO 6]:

.NEW\_TYPE = DSCSK\_DTYPE\_F;

[7 TO 15]:

BEGIN

IF .OTHER\_TYPE EQL DSCSK\_DTYPE\_G OR

.OTHER\_TYPE EQL DSCSK\_DTYPE\_H

THEN

```
:10159      10244  6
:10160      10245  6
:10161      10246  6
:10162      10247  6
:10163      10248  6
:10164      10249  6
:10165      10250  6
:10166      10251  6
:10167      10252  6
:10168      10253  6
:10169      10254  6
:10170      10255  6
:10171      10256  6
:10172      10257  6
:10173      10258  6
:10174      10259  6
:10175      10260  6
:10176      10261  6
:10177      10262  6
:10178      10263  6
:10179      10264  6
:10180      10265  6
:10181      10266  6
:10182      10267  6
:10183      10268  6
:10184      10269  6
:10185      10270  6
:10186      10271  6

      .NEW_TYPE = .OTHER_TYPE
ELSE
      .NEW_TYPE = DSC$K_DTYPE_D;
END;

[16]:
      .NEW_TYPE = DSC$K_DTYPE_D;

[17 TO 31]:
      .NEW_TYPE = DSC$K_DTYPE_H;

[OUTRANGE]:
      SIGNAL (DBG$_ILLTYPE);
TES;
RETURN;
END;

! Presently, the other type conversions involving packed decimal types
! can be handled through the hierarchy tables.

[INRANGE]:
      RETURN;

[OUTRANGE]:
      $DBG_ERROR ('DBGEVALOP\MAP_PACKED');
TES;
END;
```

```
.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
5F 50 41 4D 5C 50 4F 4C 41 56 45 47 42 44 14 0617D P.AMT: .ASCII <20>\DBGEVALOP\<92>\MAP_PACKED\
44 45 4B 43 41 50 0618C
```

```
.PSECT DBG$CODE,NOWRT, SHR, PIC,0

000C 00000 MAP_PACKED:
53 00000000G 00 9E 00002 .WORD Save R2,R3 1018
52 04 AC D0 00009 MOVAB LIB$SIGNAL, R3 1021
15 62 D1 00000 CMPL (R2), #21
01 13 00010 BEQL 1$
04 00012 RET
CF 00013 1$: CASEL OTHER TYPE, #0, #65 1022
00000041 8F 00 OC AC CF 0001C 2$: .WORD 10$-2$, -
00FF 00FF 00FF 00FF 00024 10$-2$, -
00FF 00FF 00FF 00FF 0002C 10$-2$, -
0096 0096 00FF 00FF 00034 10$-2$, -
00FF 00FF 00FF 00FF 0003C 10$-2$, -
00FF 00FF 00FF 00FF 00044 10$-2$, -
0096 00FF 00FF 00FF 0004C 10$-2$, -
00FF 00FF 0096 00FF 00054 10$-2$, -
00FF 00FF 00FF 00FF 0005C 10$-2$, -
```

D 3  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

Page 311  
(49)

00064  
0006C  
00074  
0007C  
00084  
0008C  
00094  
0009C

[illegible]

; Routine Size: 284 bytes, Routine Base: DBG\$CODE + 26BA

```
10188 10272 1
10189 10273 1
10190 10274 1
10191 10275 1
10192 10276 1
10193 10277 1
10194 10278 1
10195 10279 1
10196 10280 1
10197 10281 1
10198 10282 1
10199 10283 1
10200 10284 1
10201 10285 1
10202 10286 1
10203 10287 1
10204 10288 1
10205 10289 1
10206 10290 1
10207 10291 1
10208 10292 1
10209 10293 1
10210 10294 2
10211 10295 2
10212 10296 2
10213 10297 2
10214 10298 2
10215 10299 2
10216 10300 2
10217 10301 2
10218 10302 2
10219 10303 2
10220 10304 2
10221 10305 2
10222 10306 2
10223 10307 2
10224 10308 2
10225 10309 2
10226 10310 2
10227 10311 2
10228 10312 2
10229 10313 2
10230 10314 2
10231 10315 2
10232 10316 2
10233 10317 2
10234 10318 2
10235 10319 2
10236 10320 2
10237 10321 2
10238 10322 2
10239 10323 2
10240 10324 2
10241 10325 2
10242 10326 2
10243 10327 2
10244 10328 2
```

ROUTINE MAP\_PLI\_TYPE\_SIZE (VALDESC, TYPE, SIZE, SRC\_FLAG): NOVALUE =

FUNCTION

PL/I run-time routines do not follow the VAX calling standard. They also do not understand VMS descriptors. Consequently, this routine was developed to map dtypes to PL/I specific types, and to determine the correct size for that specific type. This information is required in the interface to a PL/I routine.

INPUTS

VALDESC - Pointer to a value descriptor.

TYPE - Address where to place the PL/I specific type.

SIZE - Address where to place the size of the type.

SRC\_FLAG - Set to TRUE to indicate this is the source,  
set to FALSE to indicate this is the target.

OUTPUTS

The PL/I specific data type and size are returned in TYPE and SIZE.

BEGIN

MAP

TYPE: REF VECTOR[1],  
SIZE: REF VECTOR[1],  
VALDESC: REF DBG\$VALDESC;

LOCAL

SIZE\_BYTE: REF VECTOR[.BYTE], : Size in byte vector  
LANGCODE, : Language code  
PICTPTR: REF VECTOR[.BYTE], : Pointer to picture representation  
PICTVAL, : Pointer to language specific encoding  
PSCALE: VECTOR[2, .BYTE]; : Digits and Scale

SIZE\_BYTE = .SIZE;

SIZE[0] = 0;

CASE VALDESC[DBG\$B\_VALUE\_DTYPE] FROM 0 TO DBG\$K\_MAXIMUM\_DTYPE OF

SET

[DSC\$K\_DTYPE\_B]:

BEGIN

TYPE[0] = DBG\$K\_PLI\_FIX\_BIN;

SIZE[0] = 7;

END;

[DSC\$K\_DTYPE\_W]:

BEGIN

TYPE[0] = DBG\$K\_PLI\_FIX\_BIN;

SIZE[0] = 15;

END;

[DSC\$K\_DTYPE\_L]:

BEGIN

TYPE[0] = DBG\$K\_PLI\_FIX\_BIN;

SIZE[0] = 31;

END;

[DSC\$K\_DTYPE\_F]:

```
:10245      10329      BEGIN
:10246      10330      TYPE[0] = DBG$K_PLI_FLO_DEC;
:10247      10331      SIZE[0] = 7;
:10248      10332      END;
:10249      10333      [DSC$K_DTYPE_D, DSC$K_DTYPE_G]:
:10250      10334      BEGIN
:10251      10335      TYPE[0] = DBG$K_PLI_FLO_DEC;
:10252      10336      SIZE[0] = 15;
:10253      10337      END;
:10254      10338      [DSC$K_DTYPE_H]:
:10255      10339      BEGIN
:10256      10340      TYPE[0] = DBG$K_PLI_FLO_DEC;
:10257      10341      SIZE[0] = 34;
:10258      10342      END;
:10259      10343      [DSC$K_DTYPE_T]:
:10260      10344      BEGIN
:10261      10345      IF .VALDESC[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT
:10262      10346      THEN
:10263      10347      BEGIN
:10264      10348      TYPE[0] = DBG$K_PLI_PIC;
:10265      10349      DBG$STA_TYP_PICT(.VALDESC[DBG$B_DHDR_TYPEID], LANGCODE,
:10266      10350      PICTPTR, PICTVAL, PSIZE);
:10267      10351      SIZE[0] = .PICTVAL;
:10268      10352      END
:10269      10353      ELSE
:10270      10354      BEGIN
:10271      10355      TYPE[0] = DBG$K_PLI_CHAR;
:10272      10356      SIZE[0] = .VALDESC[DBG$W_VALUE_LENGTH];
:10273      10357      END;
:10274      10358      END;
:10275      10359      [DSC$K_DTYPE_VT]:
:10276      10360      BEGIN
:10277      10361      LOCAL
:10278      10362      VALPTR: REF VECTOR[WORD];
:10279      10363
:10280      10364      TYPE[0] = DBG$K_PLI_CHAR_VAR;
:10281      10365      IF .SRC_FLAG
:10282      10366      THEN
:10283      10367      BEGIN
:10284      10368      VALPTR = .VALDESC[DBG$B_VALUE_POINTER];
:10285      10369      SIZE[0] = .VALPTR[0];
:10286      10370      END
:10287      10371      ELSE
:10288      10372      SIZE[0] = .VALDESC[DBG$W_VALUE_LENGTH];
:10289      10373
:10290      10374      END;
:10291      10375      [DSC$K_DTYPE_P]:
:10292      10376      BEGIN
:10293      10377      TYPE[0] = DBG$K_PLI_FIX_DEC;
:10294      10378
:10295      10379
:10296      10380      ! PLI runtime library routine expect to see 1234.1234 (8 digits
:10297      10381      ! and 4 as scale factor, normally, would be -4 as scale factor).
:10298      10382
:10299      10383      SIZE_BYTE[1] = - .VALDESC[DBG$B_VALUE_SCALE];
:10300      10384      SIZE_BYTE[0] = .VALDESC[DBG$W_VALUE_LENGTH];
:10301      10385      END;
```

```

:10302      10386      2      [DSC$K_DTYPE_V]:
:10303      10387      2      BEGIN
:10304      10388      2      TYPE[0] = DBG$K_PLI_ABIT;
:10305      10389      2      SIZE[0] = .VALDESC[DBG$W_VALUE_LENGTH];
:10306      10390      2      END;
:10307      10391      2      [DSC$K_DTYPE_VU]:
:10308      10392      2      BEGIN
:10309      10393      2      TYPE[0] = DBG$K_PLI_UBIT;
:10310      10394      2      SIZE[0] = .VALDESC[DBG$W_VALUE_LENGTH];
:10311      10395      2      END;
:10312      10396      2      [INRANGE, OUTRANGE]:
:10313      10397      2      BEGIN
:10314      10398      2      $DBG_ERROR ('DBGEVALOP\MAP_PLI_TYPE_SIZE: invalid dtype');
:10315      10399      2      END;
:10316      10400      2      TES;
:10317      10401      2
:10318      10402      1      END;

```

```

.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
5F 50 41 4D 5C 50 4F 4C 41 56 45 47 42 44 2B 06192 P.AMU: .ASCII \+DBGEVALOP\<92>\MAP_PLI_TYPE_SIZE: inv\
20 3A 45 5A 49 53 5F 45 50 59 54 5F 49 4C 50 061A1
65 70 79 74 64 20 64 69 6C 61 061B4 .ASCII \alid dtype\

```

```

.PSECT DBG$CODE,NOWRT, SHR, PIC,0
003C 00000 MAP_PLI_TYPE_SIZE:
5E      10      C2 00002      .WORD      Save R2,R3,R4,R5
53      0C      AC 00005      SUBL2      #16, SP
54      53      D0 00009      MOVL       SIZE, R3
63      D4 0000C      MOVL       R3, SIZE_BYTE
52      04      AC 0000E      CLRL       (R3)
55      14      A2 9E 00012      MOVL       VALDESC, R2
00      02      A5 8F 00016      MOVAB     20(R2), R5
0058      0058      00E1 0058      0001B 1$: CASEB     2(R5), #0, #43
0074      006E      0058      0058      00023      .WORD      2$-1$, -
008A      0082      0058      007A      0002B      15$-1$, -
0058      009A      0058      0058      00033      2$-1$, -
0058      0058      0058      0058      0003B      2$-1$, -
0058      0058      00D4      0058      00043      2$-1$, -
008A      0058      0058      0058      0004B      2$-1$, -
0058      0058      0058      0092      00053      3$-1$, -
0058      00E7      0058      0058      0005B      4$-1$, -
0058      0058      00C4      0058      00063      5$-1$, -
0058      0058      0058      0058      0006B      2$-1$, -
0058      0058      0058      0058      0006B      6$-1$, -
0058      0058      0058      0058      0006B      8$-1$, -
0058      0058      0058      0058      0006B      2$-1$, -
0058      0058      0058      0058      0006B      2$-1$, -
0058      0058      0058      0058      0006B      11$-1$, -
0058      0058      0058      0058      0006B      2$-1$, -
0058      0058      0058      0058      0006B      2$-1$, -

```

```

2$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
14$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
8$-1$,-
10$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
16$-1$,-
2$-1$,-
2$-1$,-
13$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
00000000' EF 9F 00073 2$: PUSHAB P.AMU
01 DD 00079 PUSHL #1
00028362 8F DD 0007B PUSHL #164706
00000000G 00 03 FB 00081 CALLS #3, LIB$SIGNAL
04 00088 RET
08 BC 02 D0 00089 3$: MOVL #2, @TYPE
12 11 0008D BRB 7$
08 BC 02 D0 0008F 4$: MOVL #2, @TYPE
14 11 00093 BRB 9$
08 BC 02 D0 00095 5$: MOVL #2, @TYPE
63 1F D0 00099 MOVL #31, (R3)
04 0009C RET
08 BC 05 D0 0009D 6$: MOVL #5, @TYPE
63 07 D0 000A1 7$: MOVL #7, (R3)
04 000A4 RET
08 BC 05 D0 000A5 8$: MOVL #5, @TYPE
63 0F D0 000A9 9$: MOVL #15, (R3)
04 000AC RET
08 BC 05 D0 000AD 10$: MOVL #5, @TYPE
63 22 D0 000B1 MOVL #34, (R3)
04 000B4 RET
05 06 A2 91 000B5 11$: CMPB 6(R2), #5
1E 12 000B9 BNEQ 12$
08 BC 01 D0 000BB MOVL #1, @TYPE
5E DD 000BF PUSHL SP
08 AE 9F 000C1 PUSHAB PICTVAL
10 AE 9F 000C4 PUSHAB PICTPTR
18 AE 9F 000C7 PUSHAB LANGCODE
08 A2 DD 000CA PUSHL 8(R2)
00000000G 00 05 FB 000CD CALLS #5, DBG$STA_TYP_PICT
1039
1031
1031
1031
1032
1032
1032
1032
1032
1033
1033
1031
1033
1033
1033
1031
1034
1034
1031
1034
1034
1034
```

|    |    |    |    |          |       |        |                      |  |      |
|----|----|----|----|----------|-------|--------|----------------------|--|------|
|    | 63 | 04 | AE | D0 000D4 |       | MOVL   | PICTVAL, (R3)        |  | 1035 |
|    |    |    |    | 04 000D8 |       | RET    |                      |  | 1034 |
| 08 | BC |    | 0A | D0 000D9 | 12\$: | MOVL   | #10, @TYPE           |  | 1035 |
|    |    |    | 27 | 11 000DD |       | BRB    | 17\$                 |  | 1035 |
| 08 | BC |    | 0B | D0 000DF | 13\$: | MOVL   | #11, @TYPE           |  | 1036 |
|    | 1F | 10 | AC | E9 000E3 |       | BLBC   | SRC FLAG, 17\$       |  | 1036 |
|    | 50 | 18 | A2 | D0 000E7 |       | MOVL   | 24(R2), VALPTR       |  | 1036 |
|    | 63 |    | 60 | 3C 000EB |       | MOVZWL | (VALPTR), (R3)       |  | 1036 |
|    |    |    |    | 04 000EE |       | RET    |                      |  | 1036 |
| 08 | BC |    | 04 | D0 000EF | 14\$: | MOVL   | #4, @TYPE            |  | 1037 |
| 01 | A4 | 1C | A2 | 8E 000F3 |       | MNEGB  | 28(R2), 1(SIZE_BYTE) |  | 1038 |
|    | 64 |    | 65 | 90 000F8 |       | MOVB   | (R5), (SIZE_BYTE)    |  | 1038 |
|    |    |    |    | 04 000FB |       | RET    |                      |  | 1031 |
| 08 | BC |    | 0E | D0 000FC | 15\$: | MOVL   | #14, @TYPE           |  | 1038 |
|    |    |    | 04 | 11 00100 |       | BRB    | 17\$                 |  | 1038 |
| 08 | BC |    | 0C | D0 00102 | 16\$: | MOVL   | #12, @TYPE           |  | 1039 |
|    | 63 |    | 65 | 3C 00106 | 17\$: | MOVZWL | (R5), (R3)           |  | 1039 |
|    |    |    |    | 04 00109 |       | RET    |                      |  | 1040 |

: Routine Size: 266 bytes, Routine Base: DBG\$CODE + 27D6

10320 10403 1  
10321 10404 1  
10322 10405 1  
10323 10406 1  
10324 10407 1  
10325 10408 1  
10326 10409 1  
10327 10410 1  
10328 10411 1  
10329 10412 1  
10330 10413 1  
10331 10414 1  
10332 10415 1  
10333 10416 1  
10334 10417 1  
10335 10418 1  
10336 10419 1  
10337 10420 1  
10338 10421 1  
10339 10422 1  
10340 10423 1  
10341 10424 1  
10342 10425 1  
10343 10426 1  
10344 10427 1  
10345 10428 1  
10346 10429 1  
10347 10430 1  
10348 10431 1  
10349 10432 1  
10350 10433 1  
10351 10434 1  
10352 10435 1  
10353 10436 1  
10354 10437 1  
10355 10438 1  
10356 10439 1  
10357 10440 1  
10358 10441 1  
10359 10442 1  
10360 10443 1  
10361 10444 1  
10362 10445 1  
10363 10446 1  
10364 10447 1  
10365 10448 1  
10366 10449 1  
10367 10450 1  
10368 10451 1  
10369 10452 1  
10370 10453 1  
10371 10454 1  
10372 10455 1  
10373 10456 1  
10374 10457 1  
10375 10458 1  
10376 10459 1

ROUTINE MODIFY\_PLI\_TARGET\_TYPE(OPERATOR, LEFT\_TYPE, RIGHT\_TYPE,  
NEW\_LEFT\_TYPE, NEW\_RIGHT\_TYPE, NEW\_ROOT\_INDEX,  
HIER\_TBL, HIER\_TBL\_SIZE,  
INCOMP\_TBL, INCOMP\_TBL\_SIZE,  
ROUT\_TBL, ROUT\_TBL\_SIZE) : NOVALUE =

## FUNCTION

This routine is used to adjust the target data type based on the type of the other operand for arithmetic and relational operators in PLI.

PLI has very unique type conversion rule, which differs from the other language, graph hierarchy scheme stand alone can not satisfy the needs without modification. This routine is a separate routine to serve the needs.

After the walking the graph from previous FIND\_JOIN, we know the data types we work with are valid. First we check the operator, if the operator is not one of the arithmetic/relational operators, we know the result from FIND\_JOIN is ok. Second, we check to see if one of the operands has data type V, VU, T, or VT, if not, we know the result from FIND\_JOIN is ok. Third, we modify the target based on the following tables:

|         | B,W,L   | P       | F,D,G,H | V,VU    | T,VT    |
|---------|---------|---------|---------|---------|---------|
| B,W,L   | B,W,L   | B,W,L   | F,D,G,H | L       | L       |
| P       | L       | P       | F,D,G,H | L       | P       |
| F,D,G,H | F,D,G,H | F,D,G,H | F,D,G,H | F,D,G,H | F,D,G,H |
| V,VU    | L       | L       | F,D,G,H | *1      | L       |
| T,VT    | L       | P       | F,D,G,H | L       | *2      |

\*1 - Relational operator: no change for data types.  
Arithmetic operator: L

\*2 - Relational operator: no change for data types.  
Arithmetic operator: P

Note: Unary operator does not have the problem, so this routine only works with binary operator.

Note: FIXED\_BIN, FIXED\_DEC, FLOAT\_DEC, CHAR, CHAR\_VAR, BIT, and BIT\_ALIGNED are the data types we know in PLI. (there are others ie., FLOAT\_BIN, BIT\_VAR.).

## INPUTS

OPERATOR - Token Operator.  
LEFT\_TYPE - Original left data type.  
RIGHT\_TYPE - Original right data type.  
NEW\_LEFT\_TYPE - The address of the new left data type.

```
10377 10460 1
10378 10461 1
10379 10462 1
10380 10463 1
10381 10464 1
10382 10465 1
10383 10466 1
10384 10467 1
10385 10468 1
10386 10469 1
10387 10470 1
10388 10471 1
10389 10472 1
10390 10473 1
10391 10474 1
10392 10475 1
10393 10476 1
10394 10477 1
10395 10478 1
10396 10479 1
10397 10480 1
10398 10481 1
10399 10482 1
10400 10483 1
10401 10484 1
10402 10485 1
10403 10486 1
10404 10487 1
10405 10488 1
10406 10489 1
10407 10490 1
10408 10491 1
10409 10492 1
10410 10493 1
10411 10494 1
10412 10495 1
10413 10496 1
10414 10497 1
10415 10498 1
10416 10499 1
10417 10500 1
10418 10501 1
10419 10502 1
10420 10503 1
10421 10504 1
10422 10505 1
10423 10506 1
10424 10507 1
10425 10508 1
10426 10509 1
10427 10510 1
10428 10511 1
10429 10512 1
10430 10513 1
10431 10514 1
10432 10515 1
10433 10516 1
```

```
NEW_RIGHT_TYPE - The address of the new right data type.
NEW_ROUT_INDEX - The address of the new corresponding routine index.
HIER_TBL - Hier. table.
HIER_TBL_SIZE - Hier. table size.
INCOMP_TBL - Incomp. table.
INCOMP_TBL_SIZE - Incomp. table size.
ROUT_TBL - Rout. table.
ROUT_TBL_SIZE - Rout. table size.
```

## OUTPUTS

If there is modification NEW target data type is returned along with corresponding routine index.

## BEGIN

## MAP

```
OPERATOR: REF TOKEN$ENTRY,
NEW_LEFT_TYPE: REF VECTOR[.LONG],
NEW_RIGHT_TYPE: REF VECTOR[.LONG],
NEW_ROUT_INDEX: REF VECTOR[.LONG];
```

## LOCAL

```
LEFT,
RIGHT,
GROUP1,
GROUP2;
```

! We only worry about this set of tokens. If the token is not in the set  
! simply returns.

GROUP1 = FALSE;

GROUP2 = FALSE;

```
IF .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_ADD OR
.OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_SUBTRACT OR
.OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_MULTIPLY OR
.OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_DIVIDE
```

THEN

GROUP1 = TRUE

ELSE

BEGIN

```
IF .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_EQUAL OR
.OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_NOT_EQUAL OR
.OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_GTR_THAN OR
.OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_GTR_EQUAL OR
.OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_LSS_THAN OR
.OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_LSS_EQUAL
```

THEN

GROUP2 = TRUE

ELSE

RETURN 0;

END;

```
10434 10517 2
10435 10518
10436 10519
10437 10520
10438 10521
10439 10522
10440 10523
10441 10524
10442 10525
10443 10526
10444 10527
10445 10528
10446 10529
10447 10530
10448 10531
10449 10532
10450 10533
10451 10534
10452 10535
10453 10536
10454 10537
10455 10538
10456 10539
10457 10540
10458 10541
10459 10542
10460 10543
10461 10544
10462 10545
10463 10546
10464 10547
10465 10548
10466 10549
10467 10550
10468 10551
10469 10552
10470 10553
10471 10554
10472 10555
10473 10556
10474 10557
10475 10558
10476 10559
10477 10560
10478 10561
10479 10562
10480 10563
10481 10564
10482 10565
10483 10566
10484 10567
10485 10568
10486 10569
10487 10570
10488 10571
10489 10572
10490 10573
```

```
! Smash VT to T, VU to V. As far as this routine's concern, they are
! treated the same. In this way, we reduce some data types to handle.
```

```
LEFT = .LEFT TYPE;
RIGHT = .RIGHT TYPE;
IF .LEFT EQL DSC$K_DTYPE_VT THEN LEFT = DSC$K_DTYPE_T;
IF .LEFT EQL DSC$K_DTYPE_VU THEN LEFT = DSC$K_DTYPE_V;
IF .RIGHT EQL DSC$K_DTYPE_VT THEN RIGHT = DSC$K_DTYPE_T;
IF .RIGHT EQL DSC$K_DTYPE_VU THEN RIGHT = DSC$K_DTYPE_V;
```

```
! We only worry about this set of data types. If one of the operand
! is V or T then we need to modify the target. Otherwise, simply
! returns.
```

```
IF .LEFT_TYPE NEQ DSC$K_DTYPE_V AND
    .LEFT_TYPE NEQ DSC$K_DTYPE_T AND
    .RIGHT_TYPE NEQ DSC$K_DTYPE_V AND
    .RIGHT_TYPE NEQ DSC$K_DTYPE_T
THEN
    RETURN 0;
```

```
! Filter out V op V, T op T, V op T, T op V cases.
```

```
IF (.LEFT EQL DSC$K_DTYPE_V AND .RIGHT EQL DSC$K_DTYPE_V) OR
    (.LEFT EQL DSC$K_DTYPE_T AND .RIGHT EQL DSC$K_DTYPE_T) OR
    (.LEFT EQL DSC$K_DTYPE_V AND .RIGHT EQL DSC$K_DTYPE_T) OR
    (.LEFT EQL DSC$K_DTYPE_T AND .RIGHT EQL DSC$K_DTYPE_V)
THEN
    BEGIN
```

```
! If this is relational operator, then we all set.
```

```
IF .GROUP2
THEN
    RETURN 0
```

```
! If this is arithmetic operator, modify the target.
```

```
ELSE
    BEGIN
        IF .LEFT EQL DSC$K_DTYPE_T AND .RIGHT EQL DSC$K_DTYPE_T
        THEN
            BEGIN
                NEW_LEFT_TYPE[0] = DSC$K_DTYPE_P;
                NEW_RIGHT_TYPE[0] = DSC$K_DTYPE_P;
            END
        ELSE
            BEGIN
                NEW_LEFT_TYPE[0] = DSC$K_DTYPE_L;
                NEW_RIGHT_TYPE[0] = DSC$K_DTYPE_L;
            END;
```

```
10491 10574 4
10492 10575 4
10493 10576 4
10494 10577 4
10495 10578 4
10496 10579 4
10497 10580 4
10498 10581 4
10499 10582 4
10500 10583 4
10501 10584 4
10502 10585 4
10503 10586 4
10504 10587 4
10505 10588 4
10506 10589 4
10507 10590 4
10508 10591 4
10509 10592 4
10510 10593 4
10511 10594 4
10512 10595 4
10513 10596 4
10514 10597 4
10515 10598 4
10516 10599 4
10517 10600 4
10518 10601 4
10519 10602 4
10520 10603 4
10521 10604 4
10522 10605 4
10523 10606 4
10524 10607 4
10525 10608 4
10526 10609 4
10527 10610 4
10528 10611 4
10529 10612 4
10530 10613 4
10531 10614 4
10532 10615 4
10533 10616 4
10534 10617 4
10535 10618 4
10536 10619 4
10537 10620 4
10538 10621 4
10539 10622 4
10540 10623 4
10541 10624 4
10542 10625 4
10543 10626 4
10544 10627 4
10545 10628 4
10546 10629 4
10547 10630 4

! Get the corresponding routine table index.
!
IF .MAX_DEPTH EQL FIND JOIN(
    .NEW_LEFT_TYPE[0], .NEW_RIGHT_TYPE[0],
    NEW_LEFT_TYPE[0], NEW_RIGHT_TYPE[0],
    NEW_ROUT_INDEX[0],
    0, .MAX_DEPTH,
    .HIER_TBL, .HIER_TBL_SIZE,
    .INCOMP_TBL, .INCOMP_TBL_SIZE,
    .ROUT_TBL, .ROUT_TBL_SIZE)
THEN
    SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
RETURN 0;
END;
END;

! Now we have V op (R, W, L, P, F, D, G, H) or T op (B, W, ...) cases.
!
INCR I FROM 0 TO 1 DO
    BEGIN
        LOCAL
            TYPE, OTHER_TYPE;

        IF .I EQL 0
        THEN
            BEGIN
                TYPE = .LEFT;
                OTHER_TYPE = .RIGHT;
            END
        ELSE
            BEGIN
                TYPE = .RIGHT;
                OTHER_TYPE = .LEFT;
            END;

        SELECTONE .TYPE OF
            SET
                [DSC$K_DTYPE_L]:
                BEGIN
                    NEW_LEFT_TYPE[0] = DSC$K_DTYPE_L;
                    NEW_RIGHT_TYPE[0] = DSC$K_DTYPE_L;
                END;

                [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_P]:
                BEGIN
                    ! Apparently, PLI documentation and PLI compiler have a different point
                    ! of view to decide which target to assign. This is the reason we
                    ! commented out this piece. We take PLI compiler's result.
                    IF .OTHER_TYPE EQL DSC$K_DTYPE_V
                    THEN
                        BEGIN
                            NEW_LEFT_TYPE[0] = DSC$K_DTYPE_L;
                        END
                    END
                END
            END
        END
```

| 003C 00000 MODIFY_PLI TARGET TYPE: |           |             |                               |      |
|------------------------------------|-----------|-------------|-------------------------------|------|
| 55                                 | 00000000' | EF 9E 00002 | WORD Save R2,R3,R4,R5         | 1040 |
|                                    |           | 51 7C 00009 | MOVAB MAX_DEPTH, R5           |      |
| 10                                 |           | 10 EF 0000B | CLRG GROUP1                   | 1049 |
| 06                                 |           | 50 B1 00011 | EXTZV #16, #16, @OPERATOR, R0 | 1049 |
|                                    |           | 0F 13 00014 | CMPL R0, #6                   |      |
| 07                                 |           | 50 B1 00016 | BEQL 1\$                      |      |
|                                    |           | 0A 13 00019 | CMPL R0, #7                   | 1049 |
| 08                                 |           | 50 B1 0001B | BEQL 1\$                      |      |
|                                    |           | 05 13 0001E | CMPL R0, #8                   | 1049 |
| 09                                 |           | 50 B1 00020 | BEQL 1\$                      |      |
|                                    |           |             | CMPL R0, #9                   | 1049 |

|    |    |    |    |       |      |      |                   |  |  |      |
|----|----|----|----|-------|------|------|-------------------|--|--|------|
| 51 |    | 05 | 12 | 00023 |      | BNEQ | 28                |  |  |      |
|    |    | 01 | D0 | 00025 | 18:  | MOVL | #1, GROUP1        |  |  | 1049 |
|    |    | 21 | 11 | 00028 |      | BRB  | 48                |  |  |      |
| 0D |    | 50 | B1 | 0002A | 28:  | CMPW | R0, #13           |  |  | 1050 |
|    |    | 19 | 13 | 0002D |      | BEQL | 38                |  |  |      |
| 0E |    | 50 | B1 | 0002F |      | CMPW | R0, #14           |  |  | 1050 |
|    |    | 14 | 13 | 00032 |      | BEQL | 38                |  |  |      |
| 0F |    | 50 | B1 | 00034 |      | CMPW | R0, #15           |  |  | 1050 |
|    |    | 0F | 13 | 00037 |      | BEQL | 38                |  |  |      |
| 11 |    | 50 | B1 | 00039 |      | CMPW | R0, #17           |  |  | 1050 |
|    |    | 0A | 13 | 0003C |      | BEQL | 38                |  |  |      |
| 13 |    | 50 | B1 | 0003E |      | CMPW | R0, #19           |  |  | 1050 |
|    |    | 05 | 13 | 00041 |      | BEQL | 38                |  |  |      |
| 15 |    | 50 | B1 | 00043 |      | CMPW | R0, #21           |  |  | 1050 |
|    |    | 41 | 12 | 00046 |      | BNEQ | 98                |  |  |      |
| 52 |    | 01 | D0 | 00048 | 38:  | MOVL | #1, GROUP2        |  |  | 1051 |
| 54 | 08 | AC | D0 | 0004B | 48:  | MOVL | LEFT_TYPE, LEFT   |  |  | 1052 |
| 53 | 0C | AC | D0 | 0004F |      | MOVL | RIGHT_TYPE, RIGHT |  |  | 1052 |
| 25 |    | 54 | D1 | 00053 |      | CMPL | LEFT, #37         |  |  | 1052 |
|    |    | 03 | 12 | 00056 |      | BNEQ | 58                |  |  |      |
| 54 |    | 0E | D0 | 00058 |      | MOVL | #14, LEFT         |  |  |      |
| 22 |    | 54 | D1 | 0005B | 58:  | CMPL | LEFT, #34         |  |  | 1052 |
|    |    | 03 | 12 | 0005E |      | BNEQ | 68                |  |  |      |
| 54 |    | 01 | D0 | 00060 |      | MOVL | #1, LEFT          |  |  |      |
| 25 |    | 53 | D1 | 00063 | 68:  | CMPL | RIGHT, #37        |  |  | 1052 |
|    |    | 03 | 12 | 00066 |      | BNEQ | 78                |  |  |      |
| 53 |    | 0E | D0 | 00068 |      | MOVL | #14, RIGHT        |  |  |      |
| 22 |    | 53 | D1 | 0006B | 78:  | CMPL | RIGHT, #34        |  |  | 1052 |
|    |    | 03 | 12 | 0006E |      | BNEQ | 88                |  |  |      |
| 53 |    | 01 | D0 | 00070 |      | MOVL | #1, RIGHT         |  |  |      |
| 01 | 08 | AC | D1 | 00073 | 88:  | CMPL | LEFT_TYPE, #1     |  |  | 1053 |
|    |    | 13 | 13 | 00077 |      | BEQL | 108               |  |  |      |
| 0E | 08 | AC | D1 | 00079 |      | CMPL | LEFT_TYPE, #14    |  |  | 1053 |
|    |    | 0D | 13 | 0007D |      | BEQL | 108               |  |  |      |
| 01 | 0C | AC | D1 | 0007F |      | CMPL | RIGHT_TYPE, #1    |  |  | 1053 |
|    |    | 07 | 13 | 00083 |      | BEQL | 108               |  |  |      |
| 0E | 0C | AC | D1 | 00085 |      | CMPL | RIGHT_TYPE, #14   |  |  | 1053 |
|    |    | 01 | 13 | 00089 | 98:  | BEQL | 108               |  |  |      |
|    |    |    | 04 | 0008B |      | RET  |                   |  |  |      |
|    |    | 50 | D4 | 0008C | 108: | CLRL | R0                |  |  | 1054 |
| 01 |    | 54 | D1 | 0008E |      | CMPL | LEFT, #1          |  |  |      |
|    |    | 07 | 12 | 00091 |      | BNEQ | 118               |  |  |      |
|    |    | 50 | D6 | 00093 |      | INCL | R0                |  |  |      |
| 01 |    | 53 | D1 | 00095 |      | CMPL | RIGHT, #1         |  |  |      |
|    |    | 1C | 13 | 00098 |      | BEQL | 148               |  |  |      |
| 0E |    | 54 | D1 | 0009A | 118: | CMPL | LEFT, #14         |  |  | 1054 |
|    |    | 05 | 12 | 0009D |      | BNEQ | 128               |  |  |      |
| 0E |    | 53 | D1 | 0009F |      | CMPL | RIGHT, #14        |  |  |      |
|    |    | 12 | 13 | 000A2 |      | BEQL | 148               |  |  |      |
| 05 |    | 50 | E9 | 000A4 | 128: | BLBC | R0, 138           |  |  | 1054 |
| 0E |    | 53 | D1 | 000A7 |      | CMPL | RIGHT, #14        |  |  |      |
|    |    | 0A | 13 | 000AA |      | BEQL | 148               |  |  |      |
| 0E |    | 54 | D1 | 000AC | 138: | CMPL | LEFT, #14         |  |  | 1054 |
|    |    | 27 | 12 | 000AF |      | BNEQ | 178               |  |  |      |
| 01 |    | 53 | D1 | 000B1 |      | CMPL | RIGHT, #1         |  |  |      |
|    |    | 22 | 12 | 000B4 |      | BNEQ | 178               |  |  |      |
| 01 |    | 52 | E9 | 000B6 | 148: | BLBC | GROUP2, 158       |  |  | 1055 |

|      |    |          |    |    |       |              |                       |  |      |
|------|----|----------|----|----|-------|--------------|-----------------------|--|------|
|      | OE |          | 54 | 04 | 000B9 | RET          |                       |  |      |
|      |    |          | OF | D1 | 000BA | 15\$: CMPL   | LEFT, #14             |  | 1056 |
|      | OE |          | 53 | D1 | 000BD | BNEQ         | 16\$                  |  |      |
|      |    |          | 0A | D1 | 000BF | CMPL         | RIGHT, #14            |  |      |
| 10   | BC |          | 15 | D0 | 000C2 | BNEQ         | 16\$                  |  | 1056 |
| 14   | BC |          | 15 | D0 | 000C4 | MOVL         | #21, @NEW_LEFT_TYPE   |  | 1056 |
|      |    |          | 61 | 11 | 000C8 | MOVL         | #21, @NEW_RIGHT_TYPE  |  | 1056 |
| 10   | BC |          | 08 | D0 | 000CC | BRB          | 28\$                  |  | 1057 |
| 14   | BC |          | 08 | D0 | 000CE | 16\$: MOVL   | #8, @NEW_LEFT_TYPE    |  | 1057 |
|      |    |          | 57 | D0 | 000D2 | MOVL         | #8, @NEW_RIGHT_TYPE   |  | 1058 |
|      |    |          | 52 | 11 | 000D6 | BRB          | 28\$                  |  | 1060 |
|      |    |          | 08 | D4 | 000D8 | 17\$: CLRL   | 1                     |  | 1060 |
|      | 50 |          | 54 | D0 | 000DA | 18\$: BNEQ   | 19\$                  |  | 1060 |
|      | 51 |          | 53 | D0 | 000DC | MOVL         | LEFT, TYPE            |  | 1060 |
|      |    |          | 03 | D0 | 000DF | MOVL         | RIGHT, OTHER_TYPE     |  | 1060 |
|      | 50 |          | 53 | 11 | 000E2 | BRB          | 20\$                  |  | 1060 |
|      | 08 |          | 50 | 7D | 000E4 | 19\$: MOVQ   | RIGHT, TYPE           |  | 1061 |
|      |    |          | 0A | D1 | 000E7 | 20\$: CMPL   | TYPE, #8              |  | 1061 |
| 10   | BC |          | 08 | D0 | 000EA | BNEQ         | 21\$                  |  | 1061 |
| 14   | BC |          | 08 | D0 | 000EC | MOVL         | #8, @NEW_LEFT_TYPE    |  | 1061 |
|      |    |          | 35 | D0 | 000F0 | MOVL         | #8, @NEW_RIGHT_TYPE   |  | 1061 |
|      | 06 |          | 50 | 11 | 000F4 | BRB          | 27\$                  |  | 1062 |
|      |    |          | 05 | D1 | 000F6 | 21\$: CMPL   | TYPE, #6              |  |      |
|      | 07 |          | 50 | 19 | 000F9 | BLSS         | 22\$                  |  |      |
|      |    |          | 05 | D1 | 000FB | CMPL         | TYPE, #7              |  |      |
|      | 15 |          | 50 | 15 | 000FE | BLEQ         | 23\$                  |  |      |
|      |    |          | 0A | D1 | 00100 | 22\$: CMPL   | TYPE, #21             |  |      |
| 10   | BC |          | 15 | D0 | 00103 | BNEQ         | 24\$                  |  | 1063 |
| 14   | BC |          | 15 | D0 | 00105 | 23\$: MOVL   | #21, @NEW_LEFT_TYPE   |  | 1063 |
|      |    |          | 1C | D0 | 00109 | MOVL         | #21, @NEW_RIGHT_TYPE  |  | 1061 |
|      | 0A |          | 50 | 11 | 0010D | BRB          | 27\$                  |  | 1064 |
|      |    |          | 05 | D1 | 0010F | 24\$: CMPL   | TYPE, #10             |  |      |
|      | 0B |          | 50 | 19 | 00112 | BLSS         | 25\$                  |  |      |
|      |    |          | 0A | D1 | 00114 | CMPL         | TYPE, #11             |  |      |
|      | 1B |          | 50 | 15 | 00117 | BLEQ         | 26\$                  |  |      |
|      |    |          | 0D | D1 | 00119 | 25\$: CMPL   | TYPE, #27             |  |      |
|      | 1C |          | 50 | 19 | 0011C | BLSS         | 27\$                  |  |      |
|      |    |          | 08 | D1 | 0011E | CMPL         | TYPE, #28             |  |      |
| 10   | BC |          | 50 | 14 | 00121 | BGTR         | 27\$                  |  | 1064 |
| 14   | BC |          | 50 | D0 | 00123 | 26\$: MOVL   | TYPE, @NEW_LEFT_TYPE  |  | 1064 |
|      |    |          | 50 | D0 | 00127 | MOVL         | TYPE, @NEW_RIGHT_TYPE |  | 1059 |
| AB   | 52 |          | 01 | F3 | 0012B | 27\$: AOBLEQ | #1, 18\$              |  | 1066 |
|      | 7E | 2C       | AC | 7D | 0012F | 28\$: MOVQ   | ROUT fBL, -(SP)       |  | 1066 |
|      | 7E | 24       | AC | 7D | 00133 | MOVQ         | INCOMP TBL, -(SP)     |  | 1066 |
|      | 7E | 1C       | AC | 7D | 00137 | MOVQ         | HIER TBL, -(SP)       |  | 1066 |
|      |    |          | 65 | DD | 0013B | PUSHL        | MAX_DEPTH             |  | 1066 |
|      |    |          | 7E | D4 | 0013D | CLRL         | -(SP)                 |  | 1066 |
|      | 7E | 14       | AC | 7D | 0013F | MOVQ         | NEW_RIGHT_TYPE, -(SP) |  |      |
|      |    | 10       | AC | DD | 00143 | PUSHL        | NEW_LEFT_TYPE         |  |      |
|      |    | 14       | BC | DD | 00146 | PUSHL        | @NEW_RIGHT_TYPE       |  |      |
|      |    | 10       | BC | DD | 00149 | PUSHL        | @NEW_LEFT_TYPE        |  |      |
| F678 | CF |          | 0D | FB | 0014C | CALLS        | #13, FIND_JOIN        |  |      |
|      | 50 |          | 65 | D1 | 00151 | CMPL         | MAX_DEPTH, R0         |  |      |
|      |    |          | 14 | D1 | 00154 | BNEQ         | 29\$                  |  |      |
| 7E   | 04 |          | 0C | C1 | 00156 | ADDL3        | #12, OPERATOR, -(SP)  |  | 1066 |
|      |    |          | 01 | DD | 0015B | PUSHL        | #1                    |  |      |
|      |    | 000289CA | 8F | DD | 0015D | PUSHL        | #166346               |  |      |

DBGEVALOP  
V04-000

E 4  
16-Sep-1984 00:32:25 VAX-11 Bliss-32 V4.0-742  
5-Sep-1984 21:54:24 [DEBUG.SRC]DBGEVALOP.B32;1

Page 325  
(51)

00000000G 00

03 FB 00163 CALLS #3, LIB\$SIGNAL  
04 0016A 29\$ RET

: 1067

; Routine Size: 363 bytes, Routine Base: DBG\$CODE + 28E0

;10588 10671 1

```
:10590      10672  1
:10591      10673  1
:10592      10674  1
:10593      10675  1
:10594      10676  1
:10595      10677  1
:10596      10678  1
:10597      10679  1
:10598      10680  1
:10599      10681  1
:10600      10682  1
:10601      10683  1
:10602      10684  1
:10603      10685  1
:10604      10686  1
:10605      10687  1
:10606      10688  1
:10607      10689  1
:10608      10690  1
:10609      10691  1
:10610      10692  1
:10611      10693  1
:10612      10694  1
:10613      10695  1
:10614      10696  1
:10615      10697  1
:10616      10698  1
:10617      10699  1
:10618      10700  1
:10619      10701  1
:10620      10702  1
:10621      10703  1
:10622      10704  1
:10623      10705  1
:10624      10706  1
:10625      10707  1
:10626      10708  1
:10627      10709  1
:10628      10710  1
:10629      10711  1
:10630      10712  1
:10631      10713  1
:10632      10714  1
:10633      10715  1
:10634      10716  1
:10635      10717  1
:10636      10718  1
:10637      10719  1
:10638      10720  1
:10639      10721  1
:10640      10722  1
:10641      10723  1
:10642      10724  1
:10643      10725  1
:10644      10726  1
:10645      10727  1
:10646      10728  1
```

```
ROUTINE PLI_TYPE_CONV (VALUE1, VALUE2) =

FUNCTION
    Performs PLI language-specific type conversion on the given
    descriptors.

INPUTS
    VALUE1      - DEBUG value descriptor for the source
    VALUE2      - DEBUG value descriptor for the target

OUTPUTS
    A pointer to the result descriptor is returned.

BEGIN

ENABLE
    PLI_HANDLER;

MAP
    VALUE1 : REF DBG$VALDESC,
    VALUE2 : REF DBG$VALDESC;

LOCAL
    SRC_ADDR,  DST_ADDR,
    SRC_TYPE,  DST_TYPE,
    SRC_SIZE,  DST_SIZE,
    SRC_OFFSET, DST_OFFSET;

    ! In PL/I, special conversions are required when going to/from
    ! bit-strings. PL/I bit-strings are stored in reverse order in
    ! memory; although they are mapped to type DSC$K_DTYPE_TF, this
    ! means that they cannot be converted by DBG$CVT_DX_DX, which
    ! treats bit-strings like integers.

    ! Get the source and destination pointers.
    SRC_ADDR = .VALUE1[DBG$SL_VALUE_POINTER];
    DST_ADDR = .VALUE2[DBG$SL_VALUE_POINTER];

    ! Map the current dtype to PL/I specific types; calculate the
    ! size for the type.
    MAP_PLI_TYPE_SIZE(.VALUE1, SRC_TYPE, SRC_SIZE, TRUE);

    ! Get the digit and scale info from picture data type.
    IF .VALUE1[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT AND
        .VALUE2[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_P
    THEN
        BEGIN
            LOCAL
                PICTVAL: REF VECTOR[.BYTE];

            PICTVAL = .SRC_SIZE;
```

```
:10647      10729      VALUE2[DBG$B_VALUE_DIGITS] = .PICTVAL[0];
:10648      10730
:10649      10731
:10650      10732      ! note: PLIRTL expects to see positive scale 2 for (12.34),
:10651      10733      ! normally we'll have -2 as the scale.
:10652      10734
:10653      10735      VALUE2[DBG$B_VALUE_SCALE] = - .PICTVAL[1];
:10654      10736      END;
:10655      10737
:10656      10738      MAP_PLI_TYPE_SIZE(.VALUE2, DST_TYPE, DST_SIZE, FALSE);
:10657      10739
:10658      10740
:10659      10741      ! Determine the bit offset. This should be zero for aligned types.
:10660      10742      ! The [DBG$B_VALUE_POS] field of the value descriptor could be
:10661      10743      ! garbage for packed decimal types, but PLISVRT_ANY ignores the
:10662      10744      ! offset for any types except those that might be unaligned.
:10663      10745
:10664      10746      SRC_OFFSET = 0;
:10665      10747      DST_OFFSET = 0;
:10666      10748      IF .VALUE1[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_VU
:10667      10749      THEN
:10668      10750          SRC_OFFSET = .VALUE1[DBG$B_VALUE_POS];
:10669      10751
:10670      10752      IF .VALUE2[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_VU
:10671      10753      THEN
:10672      10754          DST_OFFSET = .VALUE2[DBG$B_VALUE_POS];
:10673      10755
:10674      10756
:10675      10757      ! Call a PL/I run-time routine to do the conversion.
:10676      10758
:10677      10759      PLISVRT_ANY (SRC_ADDR, .SRC_TYPE, .SRC_SIZE, .SRC_OFFSET,
:10678      10760                  DST_ADDR, .DST_TYPE, .DST_SIZE, .DST_OFFSET);
:10679      10761
:10680      10762      RETURN .VALUE2;
:10681      10763      END;
```

| 000C 00000 PLI_TYPE_CONV: |    |      |    |    |       |        |                       |  |      |
|---------------------------|----|------|----|----|-------|--------|-----------------------|--|------|
|                           | 5E |      | 18 | C2 | 00002 | WORD   | Save R2,R3            |  | 1067 |
|                           | 6D | 007C | CF | DE | 00005 | SUBL2  | #24, SP               |  | 1068 |
|                           | 52 | 04   | AC | 7D | 0000A | MOVAL  | 4\$, (FP)             |  | 1070 |
| 14                        | AE | 18   | A2 | D0 | 0000E | MOVL   | VALUE1, R2            |  |      |
| 10                        | AE | 18   | A3 | D0 | 00013 | MOVL   | 24(R2), SRC_ADDR      |  | 1071 |
|                           |    |      | 01 | DD | 00018 | MOVL   | 24(R3), DST_ADDR      |  | 1071 |
|                           |    | 04   | AE | 9F | 0001A | PUSHL  | #1                    |  |      |
|                           |    | 0C   | AE | 9F | 0001D | PUSHAB | SRC_SIZE              |  |      |
|                           |    |      | 52 | DD | 00020 | PUSHAB | SRC_TYPE              |  |      |
| FD64                      | CF |      | 04 | FB | 00022 | PUSHL  | R2                    |  |      |
|                           | 05 | 06   | A2 | 91 | 00027 | CALLS  | #4, MAP_PLI_TYPE_SIZE |  | 1072 |
|                           |    |      | 12 | 12 | 0002B | CMPL   | 6(R2), #5             |  |      |
|                           | 15 | 16   | A3 | 91 | 0002D | BNEQ   | 1\$                   |  | 1072 |
|                           |    |      | 0C | 12 | 00031 | CMPL   | 22(R3), #21           |  |      |
|                           | 50 |      | 6E | D0 | 00033 | BNEQ   | 1\$                   |  |      |
|                           |    |      |    |    |       | MOVL   | SRC_SIZE, PICTVAL     |  | 1072 |

|           |    |    |      |       |       |              |                       |      |
|-----------|----|----|------|-------|-------|--------------|-----------------------|------|
| 1D        | A3 |    | 60   | 90    | 00036 | MOVB         | (PICTVAL), 29(R3)     | 1072 |
| 1C        | A3 | 01 | A0   | 8E    | 0003A | MNEGB        | 1(PICTVAL), 28(R3)    | 1073 |
|           |    |    | 7E   | D4    | 0003F | CLRL         | -(SP)                 | 1073 |
|           |    | 0C | AE   | 9F    | 00041 | PUSHAB       | DST_SIZE              |      |
|           |    | 14 | AE   | 9F    | 00044 | PUSHAB       | DST_TYPE              |      |
|           |    |    | 53   | DD    | 00047 | PUSHL        | R3                    |      |
| FD3D      | CF |    | 04   | FB    | 00049 | CALLS        | #4, MAP_PLI_TYPE_SIZE |      |
|           |    |    | 50   | 7C    | 0004E | CLRL         | DST_OFFSET            | 1074 |
|           | 22 | 16 | A2   | 91    | 00050 | CMPB         | 22(R2), #34           | 1074 |
|           |    |    | 04   | 12    | 00054 | BNEQ         | 2\$                   |      |
|           | 51 | 1C | A2   | D0    | 00056 | MOVL         | 28(R2), SRC_OFFSET    | 1075 |
|           | 22 | 16 | A3   | 91    | 0005A | CMPB         | 22(R3), #34           | 1075 |
|           |    |    | 04   | 12    | 0005E | BNEQ         | 3\$                   |      |
|           | 50 | 1C | A3   | D0    | 00060 | MOVL         | 28(R3), DST_OFFSET    | 1075 |
|           |    |    | 50   | DD    | 00064 | PUSHL        | DST_OFFSET            | 1076 |
|           |    | 0C | AE   | DD    | 00066 | PUSHL        | DST_SIZE              |      |
|           |    | 14 | AE   | DD    | 00069 | PUSHL        | DST_TYPE              |      |
|           |    | 1C | AE   | 9F    | 0006C | PUSHAB       | DST_ADDR              | 1075 |
|           |    |    | 51   | DD    | 0006F | PUSHL        | SRC_OFFSET            |      |
|           |    | 14 | AE   | DD    | 00071 | PUSHL        | SRC_SIZE              |      |
|           |    | 1C | AE   | DD    | 00074 | PUSHL        | SRC_TYPE              |      |
|           |    | 30 | AE   | 9F    | 00077 | PUSHAB       | SRC_ADDR              |      |
| 00000000G | 00 |    | 08   | FB    | 0007A | CALLS        | #8, PLI\$CVRT_ANY     | 1076 |
|           | 50 |    | 53   | D0    | 00081 | MOVL         | R3, R0                | 1076 |
|           |    |    | 04   | 00084 | RET   |              |                       | 1068 |
|           |    |    | 0000 | 00085 | .WORD | Save nothing |                       |      |
|           |    |    | 7E   | D4    | 00087 | CLRL         | -(SP)                 |      |
|           |    |    | 5E   | DD    | 00089 | PUSHL        | SP                    |      |
| 0000V     | 7E | 04 | AC   | 7D    | 0008B | MOVQ         | 4(AP), -(SP)          |      |
|           | CF |    | 03   | FB    | 0008F | CALLS        | #3, PLI_HANDLER       |      |
|           |    |    | 04   | 00094 | RET   |              |                       |      |

; Routine Size: 149 bytes, Routine Base: DBG\$CODE + 2A4B

10683 10764 1  
10684 10765 1  
10685 10766 1  
10686 10767 1  
10687 10768 1  
10688 10769 1  
10689 10770 1  
10690 10771 1  
10691 10772 1  
10692 10773 1  
10693 10774 1  
10694 10775 1  
10695 10776 1  
10696 10777 1  
10697 10778 1  
10698 10779 1  
10699 10780 1  
10700 10781 1  
10701 10782 1  
10702 10783 1  
10703 10784 1  
10704 10785 1  
10705 10786 1  
10706 10787 1  
10707 10788 1  
10708 10789 1  
10709 10790 1  
10710 10791 1  
10711 10792 1  
10712 10793 1  
10713 10794 1  
10714 10795 1  
10715 10796 1  
10716 10797 1  
10717 10798 1  
10718 10799 1  
10719 10800 1  
10720 10801 1  
10721 10802 1  
10722 10803 1  
10723 10804 1  
10724 10805 1  
10725 10806 1

```
ROUTINE PLI_HANDLER (SIG, MECH) =  
FUNCTION  
    This handler catches PL/I specific data type conversion errors, and  
    signals them as Debug errors. It resignals all other conditions.  
INPUTS  
    SIG - A counted vector of parameters describing the condition.  
    MECH - A counted vector of parameters from CHF.  
OUTPUTS  
  
BEGIN  
MAP  
    SIG: REF VECTOR[,LONG];  
  
    !Translate all numeric exceptions to debug's facility code.  
    !Also, translate SS$_ROPRAND to DBG$_ROPRANDF.  
SELECTONE .SIG[1] OF  
    SET  
    [SS$_INTOVF]:  
        SIGNAL (DBG$_INTOVF, 1, .DBG$_OPCODE_NAME);  
    [SS$_DECOVF]:  
        SIGNAL (DBG$_DECOVF, 1, .DBG$_OPCODE_NAME);  
    [SS$_FLTUVF, SS$_FLTUVF_F]:  
        SIGNAL (DBG$_FLTUVF, 1, .DBG$_OPCODE_NAME);  
    [SS$_FLTUND, SS$_FLTUND_F]:  
        SIGNAL (DBG$_FLTUND, 1, .DBG$_OPCODE_NAME);  
    [SS$_ROPRAND]:  
        SIGNAL (DBG$_ROPRANDF, 1, .DBG$_OPCODE_NAME);  
    [SS$_UNWIND]:  
        RETURN (SS$_RESIGNAL);  
    [OTHERWISE]:  
        SIGNAL (DBG$_PLICVTERR, 1, .DBG$_OPCODE_NAME);  
TES;  
  
SETUNWIND();  
RETURN 0;  
END;
```

! End of PLI\_HANDLER

```
000C 00000 PLI_HANDLER:  
53 00000000' EF 9E 00002 .WORD Save R2,R3  
50 04 AC D0 00009 MOVAB DBG$_OPCODE_NAME, R3  
50 04 C0 00000 MOVL SIG, R0  
52 60 D0 00010 ADDL2 #4, R0  
8F 52 D1 00013 MOVL (R0), R2  
0C 12 0001A CMPL R2, #1148  
63 DD 0001C BNEQ 1$  
PUSHL DBG$_OPCODE_NAME
```

1076  
1078  
1078  
1078

|           |    |          |                  |                           |      |
|-----------|----|----------|------------------|---------------------------|------|
|           |    | 000286A3 | 01 DD 0001E      | PUSHL #1                  |      |
|           |    |          | 8F DD 00020      | PUSHL #165539             |      |
| 000004A4  | 8F |          | 7F 11 00026      | BRB 9\$                   |      |
|           |    |          | 52 D1 00028 1\$: | CMPL R2, #1188            | 1079 |
|           |    |          | 0C 12 0002F      | BNEQ 2\$                  |      |
|           |    |          | 63 DD 00031      | PUSHL DBG\$GL_OPCODE_NAME | 1079 |
|           |    | 00028A3A | 01 DD 00033      | PUSHL #1                  |      |
|           |    |          | 8F DD 00035      | PUSHL #166458             |      |
| 0000048C  | 8F |          | 6A 11 00038      | BRB 9\$                   |      |
|           |    |          | 52 D1 0003D 2\$: | CMPL R2, #1164            | 1079 |
|           |    |          | 09 13 00044      | BEQL 3\$                  |      |
| 000004B4  | 8F |          | 52 D1 00046      | CMPL R2, #1204            |      |
|           |    |          | 0C 12 0004D      | BNEQ 4\$                  |      |
|           |    |          | 63 DD 0004F 3\$: | PUSHL DBG\$GL_OPCODE_NAME | 1079 |
|           |    |          | 01 DD 00051      | PUSHL #1                  |      |
|           |    | 00028A02 | 8F DD 00053      | PUSHL #166402             |      |
|           |    |          | 4C 11 00059      | BRB 9\$                   |      |
| 0000049C  | 8F |          | 52 D1 0005B 4\$: | CMPL R2, #1180            | 1079 |
|           |    |          | 09 13 00062      | BEQL 5\$                  |      |
| 000004C4  | 8F |          | 52 D1 00064      | CMPL R2, #1220            |      |
|           |    |          | 0C 12 0006B      | BNEQ 6\$                  |      |
|           |    |          | 63 DD 0006D 5\$: | PUSHL DBG\$GL_OPCODE_NAME | 1079 |
|           |    |          | 01 DD 0006F      | PUSHL #1                  |      |
|           |    | 0002869B | 8F DD 00071      | PUSHL #165531             |      |
|           |    |          | 2E 11 00077      | BRB 9\$                   |      |
| 00000454  | 8F |          | 52 D1 00079 6\$: | CMPL R2, #1108            | 1079 |
|           |    |          | 0C 12 00080      | BNEQ 7\$                  |      |
|           |    |          | 63 DD 00082      | PUSHL DBG\$GL_OPCODE_NAME | 1079 |
|           |    |          | 01 DD 00084      | PUSHL #1                  |      |
|           |    | 00028A0A | 8F DD 00086      | PUSHL #166410             |      |
|           |    |          | 19 11 0008C      | BRB 9\$                   |      |
| 00000920  | 8F |          | 52 D1 0008E 7\$: | CMPL R2, #2336            | 1079 |
|           |    |          | 06 12 00095      | BNEQ 8\$                  |      |
|           | 50 | 0918     | 8F 3C 00097      | MOVZWL #2328, R0          | 1079 |
|           |    |          | 04 0009C         | RET                       |      |
|           |    |          | 63 DD 0009D 8\$: | PUSHL DBG\$GL_OPCODE_NAME | 1080 |
|           |    |          | 01 DD 0009F      | PUSHL #1                  |      |
|           |    | 00028260 | 8F DD 000A1      | PUSHL #164448             |      |
| 00000000G | 00 |          | 03 FB 000A7 9\$: | CALLS #3, LIB\$SIGNAL     |      |
|           |    |          | 7E 7C 000AE      | CLRQ -(SP)                | 1080 |
| 00000000G | 00 |          | 02 FB 000B0      | CALLS #2, SYS\$UNWIND     |      |
|           |    |          | 50 D4 000B7      | CLRL R0                   | 1080 |
|           |    |          | 04 000B9         | RET                       | 1080 |

; Routine Size: 186 bytes, Routine Base: DBG\$CODE + 2AE0

```
10727 10807 1
10728 10808 1
10729 10809 1
10730 10810 1
10731 10811 1
10732 10812 1
10733 10813 1
10734 10814 1
10735 10815 1
10736 10816 1
10737 10817 1
10738 10818 1
10739 10819 1
10740 10820 1
10741 10821 1
10742 10822 1
10743 10823 1
10744 10824 1
10745 10825 1
10746 10826 1
10747 10827 1
10748 10828 1
10749 10829 1
10750 10830 1
10751 10831 1
10752 10832 1
10753 10833 1
10754 10834 1
10755 10835 1
10756 10836 1
10757 10837 1
10758 10838 1
10759 10839 1
10760 10840 1
10761 10841 1
10762 10842 1
10763 10843 1
10764 10844 1
10765 10845 1
10766 10846 1
10767 10847 1
10768 10848 1
10769 10849 1
10770 10850 1
10771 10851 1
10772 10852 1
10773 10853 1
10774 10854 1
10775 10855 1
10776 10856 1
10777 10857 1
10778 10858 1
10779 10859 1
10780 10860 1
10781 10861 1
10782 10862 1
10783 10863 1
```

ROUTINE TYPEID\_CHECK\_ENUM(TYPEID1, TYPEID2) =

## FUNCTION

This routine performs typeid check on the given TYPEIDs for enumeration data types.

## INPUTS

TYPEID1 - TYPEID RST entry pointer.

TYPEID2 - TYPEID RST entry pointer.

## OUTPUTS

Returned Status: TRUE or FALSE.

## BEGIN

## MAP

TYPEID1: REF RST\$ENTRY, ! Typeid pointer to RST entry  
TYPEID2: REF RST\$ENTRY; ! Typeid pointer to RST entry

## LOCAL

STATUS; ! Return status

IF .TYPEID1 EQL 0 OR .TYPEID2 EQL 0

## THEN

RETURN FALSE;

IF .TYPEID1 EQL .TYPEID2

## THEN

STATUS = TRUE

## ELSE

## BEGIN

IF .TYPEID1[RST\$B\_FCODE] EQL .TYPEID2[RST\$B\_FCODE]

## THEN

## BEGIN

IF .TYPEID1[RST\$B\_FCODE] EQL RST\$K\_TYPE\_ENUM

## THEN

## BEGIN

IF .TYPEID1[RST\$L\_DSTPTR] EQL .TYPEID2[RST\$L\_DSTPTR]

## THEN

STATUS = TRUE

## ELSE

STATUS = FALSE;

## END

## ELSE

STATUS = FALSE;

## END

## ELSE

STATUS = FALSE;

```
:10784      10864      2
:10785      10865      2
:10786      10866      2
:10787      10867      2
:10788      10868      1
```

```
END:
RETURN .STATUS;
END;
```

```
0004 00000 TYPEID_CHECK_ENUM:
      51      04      AC      D0      00002      .WORD      Save R2
      2D      13      00006      MOVL      TYPEID1, R1
      0B      AC      D5      00008      BEQL      4$
      28      13      00008      TSTL      TYPEID2
      50      0B      AC      D0      0000D      BEQL      4$
      50      51      D1      00011      MOVL      TYPEID2, R0
      14      13      00014      CMPL      R1, R0
      18      A0      1B      A1      91      00016      BEQL      1$
      12      12      00018      CMPB      24(R1), 24(R0)
      04      1B      A1      91      0001D      BNEQ      2$
      0C      A0      0C      A1      D1      00021      CMPB      24(R1), #4
      05      12      00028      BNEQ      2$
      52      01      D0      0002A      CMPL      12(R1), 12(R0)
      02      11      0002D      BNEQ      2$
      52      D4      0002F      1$:      MOVL      #1, STATUS
      52      D0      00031      2$:      BRB      3$
      04      00034      3$:      CLRL      STATUS
      50      D4      00035      4$:      MOVL      STATUS, R0
      04      00037      RET
      RET      R0
```

; Routine Size: 56 bytes, Routine Base: DBG\$CODE + 2B9A

```
:10790      10869      1
:10791      10870      1
:10792      10871      1
:10793      10872      1
:10794      10873      1
:10795      10874      1
:10796      10875      1
:10797      10876      1
:10798      10877      1
:10799      10878      1
:10800      10879      1
:10801      10880      1
:10802      10881      1
:10803      10882      1
:10804      10883      1
:10805      10884      1
:10806      10885      1
:10807      10886      1
:10808      10887      1
:10809      10888      1
:10810      10889      1
:10811      10890      1
:10812      10891      1
:10813      10892      2
:10814      10893      2
:10815      10894      2
:10816      10895      2
:10817      10896      2
:10818      10897      2
:10819      10898      2
:10820      10899      2
:10821      10900      2
:10822      10901      2
:10823      10902      2
:10824      10903      2
:10825      10904      2
:10826      10905      2
:10827      10906      2
:10828      10907      2
:10829      10908      2
:10830      10909      2
:10831      10910      2
:10832      10911      2
:10833      10912      2
:10834      10913      2
:10835      10914      2
:10836      10915      2
:10837      10916      2
:10838      10917      2
:10839      10918      2
:10840      10919      2
:10841      10920      2
:10842      10921      2
:10843      10922      2
:10844      10923      2
:10845      10924      2
:10846      10925      2
```

ROUTINE TYPEID\_CHECK\_SET(TYPEID1, TYPEID2, FCODE1, FCODE2, DTYPE1, DTYPE2) =

## FUNCTION

This routine performs typeid check on the given TYPEIDs for set data types.

## INPUTS

TYPEID1 - TYPEID RST entry pointer.

TYPEID2 - TYPEID RST entry pointer.

FCODE1 - Fcode

FCODE2 - Fcode

DTYPE1 - Data type

DTYPE2 - Data type

## OUTPUTS

Returned Status: TRUE or FALSE.

## BEGIN

## MAP

TYPEID1: REF RST\$ENTRY,

TYPEID2: REF RST\$ENTRY;

! Typeid pointer to RST entry

! Typeid pointer to RST entry

## LOCAL

STATUS;

! Return status

IF .TYPEID1 EQL .TYPEID2

THEN

STATUS = TRUE

ELSE

BEGIN

IF .FCODE1 EQL .FCODE2

THEN

BEGIN

IF .FCODE1 EQL RST\$K\_TYPE\_ATOMIC

THEN

BEGIN

IF .DTYPE1 EQL .DTYPE2

THEN

BEGIN

IF .DTYPE1 EQL DSC\$K\_DTYPE\_TF OR

.DTYPE1 EQL DSC\$K\_DTYPE\_L OR

.DTYPE1 EQL DSC\$K\_DTYPE\_LU OR

.DTYPE1 EQL DSC\$K\_DTYPE\_T

THEN

STATUS = TRUE

ELSE

STATUS = FALSE;

END

```

ELSE
    STATUS = FALSE;
END
ELSE
    BEGIN
        IF .FCODE1 EQL RST$K_TYPE_ENUM
        THEN
            BEGIN
                IF .TYPEID1[RST$L_DSTPTR] EQL .TYPEID2[RST$L_DSTPTR]
                THEN
                    STATUS = TRUE
                ELSE
                    STATUS = FALSE;
                END
            END
        ELSE
            STATUS = FALSE;
        END;
    END
ELSE
    STATUS = FALSE;
END;
END;
RETURN .STATUS;
END;
```

|    |    | 000C 00000 |    | TYPEID_CHECK SET: |       |                     |
|----|----|------------|----|-------------------|-------|---------------------|
|    | 52 | 04         | AC | D0                | 00002 | WORD Save R2,R3     |
|    | 51 | 08         | AC | D0                | 00006 | MOVL TYPEID1, R2    |
|    | 51 |            | 52 | D1                | 0000A | MOVL TYPEID2, R1    |
|    |    |            | 38 | 13                | 0000D | CMPL R2, R1         |
| 10 | AC | 0C         | AC | D1                | 0000F | BEQL 3\$            |
|    |    |            | 36 | 12                | 00014 | CMPL FCODE1, FCODE2 |
|    | 02 | 0C         | AC | D1                | 00016 | BNEQ 4\$            |
|    |    |            | 1E | 12                | 0001A | CMPL FCODE1, #2     |
|    | 50 | 14         | AC | D0                | 0001C | BNEQ 1\$            |
| 18 | AC |            | 50 | D1                | 00020 | MOVL DTYPE1, R0     |
|    |    |            | 26 | 12                | 00024 | CMPL R0, DTYPE2     |
|    | 28 |            | 50 | D1                | 00026 | BNEQ 4\$            |
|    |    |            | 1C | 13                | 00029 | CMPL R0, #40        |
|    | 08 |            | 50 | D1                | 0002B | BEQL 3\$            |
|    |    |            | 17 | 13                | 0002E | CMPL R0, #8         |
|    | 04 |            | 50 | D1                | 00030 | BEQL 3\$            |
|    |    |            | 12 | 13                | 00033 | CMPL R0, #4         |
|    |    |            |    |                   |       | BEQL 3\$            |

DBGEVALOP  
V04-000

B 5  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32;1

Page 335  
(55)

|    |    |    |       |       |       |      |                |   |      |
|----|----|----|-------|-------|-------|------|----------------|---|------|
| 0E |    | 50 | D1    | 00035 |       | CMPL | R0, #14        | : | 1091 |
|    |    | 0B | 11    | 00038 |       | BRB  | 2\$            | : |      |
| 04 | 0C | AC | D1    | 0003A | 1\$:  | CMPL | FCODE1, #4     | : | 1093 |
|    |    | 0C | 12    | 0003E |       | BNEQ | 4\$            | : |      |
| 0C | A1 | 0C | A2    | D1    | 00040 | CMPL | 12(R2), 12(R1) | : | 1093 |
|    |    | 05 | 12    | 00045 | 2\$:  | BNEQ | 4\$            | : |      |
| 53 |    | 01 | D0    | 00047 | 3\$:  | MOVL | #1, STATUS     | : | 1093 |
|    |    | 02 | 11    | 0004A |       | BRB  | 5\$            | : |      |
|    |    | 53 | D4    | 0004C | 4\$:  | CLRL | STATUS         | : | 1095 |
| 50 |    | 53 | D0    | 0004E | 5\$:  | MOVL | STATUS, R0     | : | 1095 |
|    |    | 04 | 00051 |       |       | RET  |                | : | 1095 |

; Routine Size: 82 bytes, Routine Base: DBG\$CODE + 2BD2

:10880 10959 1

```
:10882      10960      1
:10883      10961      1
:10884      10962      1
:10885      10963      1
:10886      10964      1
:10887      10965      1
:10888      10966      1
:10889      10967      1
:10890      10968      1
:10891      10969      1
:10892      10970      1
:10893      10971      1
:10894      10972      1
:10895      10973      1
:10896      10974      1
:10897      10975      1
:10898      10976      1
:10899      10977      1
:10900      10978      1
:10901      10979      1
:10902      10980      1
:10903      10981      1
:10904      10982      1
:10905      10983      1
:10906      10984      1
:10907      10985      1
:10908      10986      1
:10909      10987      1
:10910      10988      1
:10911      10989      1
:10912      10990      1
:10913      10991      1
:10914      10992      1
:10915      10993      1
:10916      10994      1
:10917      10995      1
:10918      10996      1
:10919      10997      1
:10920      10998      1
:10921      10999      1
:10922      11000      1
:10923      11001      1
:10924      11002      1
:10925      11003      1
:10926      11004      1
:10927      11005      1
:10928      11006      1
```

```
ROUTINE TYPEID_RANGE_CHECK_ENUM(VAL_DESC, TYPEID) =
FUNCTION
    This routine takes given typeid in the value descriptor and
    performs the value range check for enumeration type.
INPUTS
    VAL_DESC      - Pointer to value descriptor.
    TYPEID        - Typeid of the data type.
OUTPUTS
    Return True to indicate the value is in the range else
    return false.
BEGIN
MAP
    TYPEID: REF RST$ENTRY,
    VAL_DESC: REF DBG$VALDESC;
LOCAL
    ADR_KIND,                ! Address kind
    ADR_PTRS: VECTOR[3, LONG], ! Address value
    NELTS,                   ! Number of elements
    NLTVEC_PTR: REF VECTOR[, LONG], ! A vector of RST entries for the elements
    SIZE;

IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM
THEN
    $DBG_ERROR('DBGEVALOP\TYPEID_RANGE_CHECK_ENUM');
DBG$STA_TYP_ENUM(.TYPEID, NELTS, NLTVEC_PTR, SIZE);
INCR I FROM 0 TO .NELTS - 1 DO
    BEGIN
        DBG$STA_SYMVALUE(.NLTVEC_PTR[I], ADR_PTRS, ADR_KIND);
        IF .(.ADR_PTRS[0]) < .ADR_PTRS[1], .SIZE, 0 > EQL
            .(.VAL_DESC[DBG$L_VALUE_POINTER])
        THEN
            RETURN TRUE;
    END;
RETURN FALSE;
END;
```

.PSECT DBG\$PLIT, NOWRT, SHR, PIC, 0

```
45 50 59 54 5C 50 4F 4C 41 56 45 47 42 44 21 061BE P.AMV: .ASCII \!DBGEVALOP\<92>\TYPEID_RANGE_CHECK_ENUM\
5F 4B 43 45 48 43 5F 45 47 4E 41 52 5F 44 49 061CD
                                4D 55 4E 45 061DC
```

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

000C 00000 TYPEID\_RANGE CHECK\_ENUM:

|           |    |          |    |       |       |        |                                  |      |
|-----------|----|----------|----|-------|-------|--------|----------------------------------|------|
|           | 5E |          | 1C | C2    | 00002 | .WORD  | Save R2,R3                       | 1096 |
|           | 52 | 08       | AC | D0    | 00005 | SUBL2  | #28, SP                          |      |
|           | 04 | 18       | A2 | 91    | 00009 | MOVL   | TYPEID, R2                       | 1099 |
|           |    |          | 15 | 13    | 0000D | CMPB   | 24(R2), #4                       |      |
|           |    | 00000000 | EF | 9F    | 0000F | BEQL   | 1\$                              |      |
|           |    |          | 01 | DD    | 00015 | PUSHAB | P.AMV                            | 1099 |
|           |    | 00028362 | 8F | DD    | 00017 | PUSHL  | #1                               |      |
| 00000000G | 00 |          | 03 | FB    | 0001D | PUSHL  | #164706                          |      |
|           |    |          | 5E | DD    | 00024 | CALLS  | #3, LIB\$SIGNAL                  |      |
|           |    | 08       | AE | 9F    | 00026 | PUSHL  | SP                               | 1099 |
|           |    | 10       | AE | 9F    | 00029 | PUSHAB | NLTVECPTTR                       |      |
|           |    |          | 52 | DD    | 0002C | PUSHAB | NELTS                            |      |
| 00000000G | 00 |          | 04 | FB    | 0002E | PUSHL  | R2                               |      |
|           | 53 | 04       | AC | D0    | 00035 | CALLS  | #4, DBG\$STA_TYP_ENUM            |      |
|           | 52 |          | 01 | CE    | 00039 | MOVL   | VAL_DESC, R3                     | 1099 |
|           |    |          | 22 | 11    | 0003C | MNEGL  | #1, -1                           |      |
|           |    | 0C       | AE | 9F    | 0003E | BRB    | 3\$                              |      |
|           |    | 14       | AE | 9F    | 00041 | PUSHAB | ADR_KIND                         | 1099 |
|           |    | 0C       | BE | 42    | DD    | PUSHAB | ADR_PTRS                         |      |
|           |    |          | 03 | FB    | 00048 | PUSHL  | @NLTVECPTTR[1]                   |      |
| 50        | 10 | BE       | 14 | AE    | EF    | CALLS  | #3, DBG\$STA_SYMVALUE            |      |
|           |    | 18       | 50 | D1    | 00056 | EXTZV  | ADR_PTRS+4, -SIZE, @ADR_PTRS, R0 | 1099 |
|           |    |          | 04 | 12    | 0005A | CMPB   | R0, -@24(R3)                     | 1099 |
|           |    | 50       | 01 | D0    | 0005C | BNEQ   | 3\$                              |      |
|           |    |          | 04 | 0005F | MOVL  | #1, R0 |                                  | 1100 |
|           |    | D9       | 52 | 08    | AE    | RET    |                                  |      |
|           |    |          | 50 | D4    | 00065 | AOBLSS | NELTS, 1, 2\$                    | 1099 |
|           |    |          | 04 | 00067 | CLRL  | R0     |                                  | 1100 |
|           |    |          |    |       | RET   |        |                                  | 1100 |

; Routine Size: 104 bytes. Routine Base: DBG\$CODE + 2C24

```
:10930 11007 1
:10931 11008 1
:10932 11009 1
:10933 11010 1
:10934 11011 1
:10935 11012 1
:10936 11013 1
:10937 11014 1
:10938 11015 1
:10939 11016 1
:10940 11017 1
:10941 11018 1
:10942 11019 1
:10943 11020 1
:10944 11021 1
:10945 11022 1
:10946 11023 1
:10947 11024 1
:10948 11025 1
:10949 11026 1
:10950 11027 1
:10951 11028 1
:10952 11029 1
:10953 11030 1
:10954 11031 1
:10955 11032 1
:10956 11033 1
:10957 11034 1
:10958 11035 1
:10959 11036 1
:10960 11037 1
:10961 11038 1
:10962 11039 1
:10963 11040 1
:10964 11041 1
:10965 11042 1
:10966 11043 1
:10967 11044 1
:10968 11045 1
:10969 11046 1
:10970 11047 1
:10971 11048 1
:10972 11049 1
:10973 11050 1
:10974 11051 1
:10975 11052 1
:10976 11053 1
:10977 11054 1
:10978 11055 1
:10979 11056 1
:10980 11057 1
:10981 11058 1
:10982 11059 1
:10983 11060 1
:10984 11061 1
:10985 11062 1
:10986 11063 1
```

```
ROUTINE TYPEID_RANGE_CHECK_SUBRNG(VAL_DESC, TYPEID) =
FUNCTION
    This routine takes given typeid in the value descriptor and
    performs the value range check for subrange type.
INPUTS
    VAL_DESC      - Pointer to value descriptor.
    TYPEID        - Typeid of the data type. This is not
                   necessary from the VAL_DESC directly.
OUTPUTS
    Return True to indicate the value is in the range else
    return false.

BEGIN
MAP
    VAL_DESC: REF DBG$VALDESC,
    TYPEID: REF RST$ENTRY;

LOCAL
    LENGTH,                ! Length
    HIGHPTR,               ! High range value
    LOWPTR,                ! Low range value
    PARENT_TYPE,           ! Parent typeid
    SETVAL: VECTOR[8, LONG],
    SETVALUE: REF BITVECTOR[],
    SET_VAL_DESC: REF VECTOR[, LONG],
    SIZE;

IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_SUBRNG
THEN
    $DBG_ERROR('DBGEVALOP\TYPEID_RANGE_CHECK_SUBRNG');

DBG$STA_TYP_SUBRNG(.TYPEID, PARENT_TYPE, LOWPTR, HIGHPTR, SIZE);
IF .VAL_DESC[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_SET
THEN
    BEGIN
        LENGTH = (.VAL_DESC[DBG$W_VALUE_LENGTH] - 1) / 4 + 1;
        INCR I FROM 0 TO .LENGTH - 1 DO
            SETVAL[I] = %X'FFFFFFFF';
        SETVALUE = SETVAL[0];
        INCR I FROM .LOWPTR TO ..HIGHPTR DO
            SETVALUE[I] = 0;
        SET_VAL_DESC = .VAL_DESC[DBG$L_VALUE_POINTER];
        INCR I FROM 0 TO .LENGTH - 1 DO
            BEGIN
                SETVAL[I] = .SET_VAL_DESC[I] AND .SETVALUE[I];
                IF .SETVAL[I] NEQ 0 THEN RETURN FALSE;
            END;
    END;

    RETURN TRUE;
END
```

```
:10987      11064 2
:10988      11065 3
:10989      11066 4
:10990      11067 5
:10991      11068 6
:10992      11069 7
:10993      11070 8
:10994      11071 9
:10995      11072 10
:10996      11073 11
:10997      11074 12

ELSE
BEGIN
IF (..HIGHPTR GEQ ..VAL_DESC[DBG$L_VALUE_POINTER]) AND
(..LOWPTR LEQ ..VAL_DESC[DBG$L_VALUE_POINTER])
THEN
RETURN TRUE
ELSE
RETURN FALSE;
END;
END;
```

```
.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
45 50 59 54 5C 50 4F 4C 41 56 45 47 42 44 23 061E0 P.AMW: .ASCII \#DBGEVALOP\<92>\TYPEID_RANGE_CHECK_SUBR\
5F 4B 43 45 48 43 5F 45 47 4E 41 52 5F 44 49 061EF
52 42 55 53 061FE
47 4E 06202 .ASCII \NG\
```

```
.PSECT DBG$CODE,NOWRT, SHR, PIC,0
000C 00000 TYPEID_RANGE_CHECK_SUBRNG:
SE 30 C2 00002 .WORD Save R2,R3 1100
52 08 AC D0 00005 SUBL2 #48, SP 1104
09 18 A2 91 00009 MOVL TYPEID, R2
15 13 0000D CMPB 24(R2), #9
00000000 EF 9F 0000F BEQL 1$ 1104
01 DD 00015 PUSHAB P.AMW
8F DD 00017 PUSHL #1
00000000G 00 00028362 8F DD 00017 PUSHL #164706
03 FB 0001D CALLS #3, LIB$SIGNAL 1104
SE DD 00024 1$: PUSHL SP
08 AE 9F 00026 PUSHAB HIGHPTR
10 AE 9F 00029 PUSHAB LOWPTR
18 AE 9F 0002C PUSHAB PARENT_TYPE
00000000G 00 05 FB 00031 PUSHL R2
52 04 AC D0 00038 CALLS #5, DBG$STA_TYP_SUBRNG 1104
08 06 A2 91 0003C MOVL VAL_DESC, R2
47 12 00040 CMPB 6(R2), #8
50 14 A2 3C 00042 BNEQ 8$ 1104
50 D7 00046 MOVZWL 20(R2), R0
04 C6 00048 DECL R0
50 D6 0004B DIVL2 #4, R0
51 01 CE 0004D INCL LENGTH 1104
05 11 00050 MNEGL #1, 1
10 AE41 01 CE 00052 2$: MNEGL #1, SETVAL[1] 1105
F7 51 50 F2 00057 3$: AOBLS LENGTH, 1, 2$
51 08 BE 01 9E 0005B MOVAB SETVAL, SETVALUE 1105
00 63 01 C3 0005F SUBL3 #1, @LOWPTR, 1 1105
F7 51 04 51 E5 00066 4$: BRB 5$
BE F3 0006A 5$: BBCC 1, (SETVALUE), 5$
AOBLEQ @HIGHPTR, 1, 4$
```

|         |    |      |    |       |       |        |                        |      |
|---------|----|------|----|-------|-------|--------|------------------------|------|
|         | 52 | 18   | A2 | D0    | 0006F | MOVL   | 24(R2), SET_VAL_DESC   | 1105 |
|         | 51 |      | 01 | CE    | 00073 | MNEGL  | #1, I                  | 1105 |
|         |    |      | 0B | 11    | 00076 | BRB    | 7\$                    |      |
|         | 53 | 6241 | D2 | 00078 | 6\$:  | MCOML  | (SET_VAL_DESC)[I], R3  |      |
| 10 AE41 |    |      | 53 | CA    | 0007C | BICL2  | R3, SETVAL[I]          |      |
|         |    |      | 18 | 12    | 00081 | BNEQ   | 10\$                   | 1105 |
| F1      | 51 |      | 50 | F2    | 00083 | AOBLSS | LENGTH, I, 6\$         | 1105 |
|         |    |      | 0E | 11    | 00087 | BRB    | 9\$                    | 1106 |
| 18 B2   | 04 |      | BE | D1    | 00089 | 8\$:   | CMPL @HIGHPTR, @24(R2) | 1106 |
|         |    |      | 0B | 19    | 0008E | BLSS   | 10\$                   |      |
| 18 B2   | 08 |      | BE | D1    | 00090 | CMPL   | @LOWPTR, @24(R2)       | 1106 |
|         |    |      | 04 | 14    | 00095 | BGTR   | 10\$                   |      |
|         | 50 |      | 01 | D0    | 00097 | 9\$:   | MOVL #1, R0            | 1107 |
|         |    |      |    | 04    | 0009A | RET    |                        |      |
|         |    |      | 50 | D4    | 0009B | 10\$:  | CLRL R0                | 1107 |
|         |    |      |    | 04    | 0009D | RET    |                        |      |

: Routine Size: 158 bytes, Routine Base: DBG\$CODE + 2C8C

:10998 11075 1  
:10999 11076 0 END ELUDOM

.EXTRN LIB\$SIGNAL, SYSSUNWIND

## PSECT SUMMARY

| Name        | Bytes | Attributes    |    |        |        |      |      |      |               |
|-------------|-------|---------------|----|--------|--------|------|------|------|---------------|
| DBG\$PLIT   | 25092 | NOVEC, NOWRT, | RD | EXE,   | SHR,   | LCL, | REL, | CON, | PIC, ALIGN(0) |
| DBG\$GLOBAL | 4     | NOVEC, WRT,   | RD | NOEXE, | NOSHR, | LCL, | REL, | CON, | PIC, ALIGN(2) |
| DBG\$OWN    | 56    | NOVEC, WRT,   | RD | NOEXE, | NOSHR, | LCL, | REL, | CON, | PIC, ALIGN(2) |
| DBG\$CODE   | 11562 | NOVEC, NOWRT, | RD | EXE,   | SHR,   | LCL, | REL, | CON, | PIC, ALIGN(0) |

## Library Statistics

| File                                     | Total | Symbols<br>Loaded | Percent | Pages<br>Mapped | Processing<br>Time |
|--|-------|-------------------|---------|-----------------|--------------------|
| _\$255\$DUA28:[SYSLIB]LIB.L32;1          | 18619 | 56                | 0       | 1000            | 00:01.9            |
| _\$255\$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1  | 32    | 3                 | 9       | 7               | 00:00.1            |
| _\$255\$DUA28:[DEBUG.OBJ]DBGLIB.L32;1    | 1545  | 518               | 33      | 97              | 00:01.9            |
| _\$255\$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1 | 418   | 24                | 5       | 31              | 00:00.3            |
| _\$255\$DUA28:[DEBUG.OBJ]DBGMSG.L32;1    | 386   | 17                | 4       | 22              | 00:00.3            |

: Information: 1  
: Warnings: 0

DBGEVALOP  
V04-000

H 5  
16-Sep-1984 00:32:25  
5-Sep-1984 21:54:24

VAX-11 BLISS-32 V4.0-742  
[DEBUG.SRC]DBGEVALOP.B32;1

Page 341  
(57)

; Errors: 0

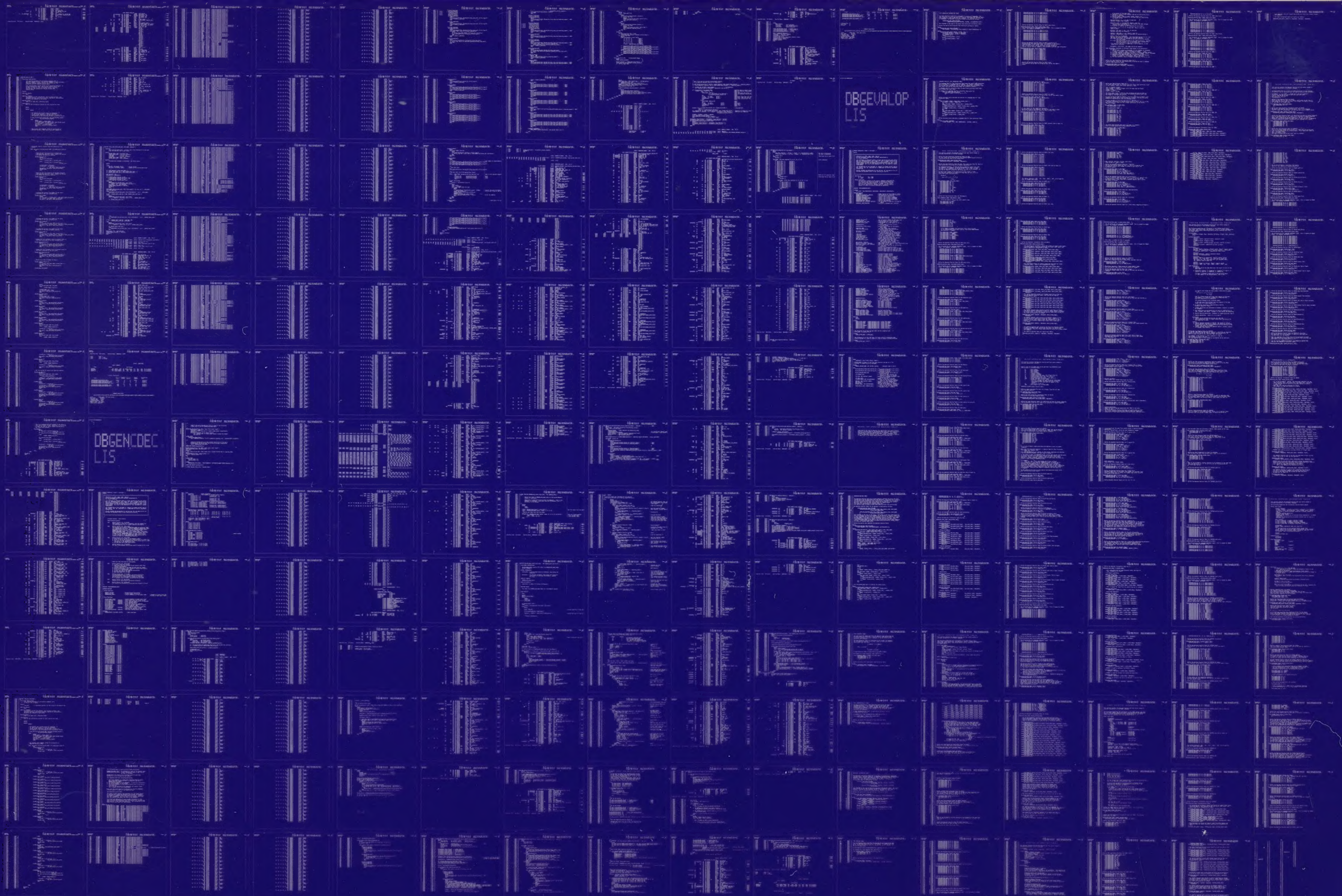
; COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DBGEVALOP/OBJ=OBJ\$:DBGEVALOP MSRC\$:DBGEVALOP/UPDATE=(ENH\$:DBGEVALOP)

; Size: 11562 code + 25152 data bytes  
; Run Time: 08:19.9  
; Elapsed Time: 26:29.6  
; Lines/CPU Min: 1329  
; Lexemes/CPU-Min: 53361  
; Memory Used: 1211 pages  
; Compilation Complete

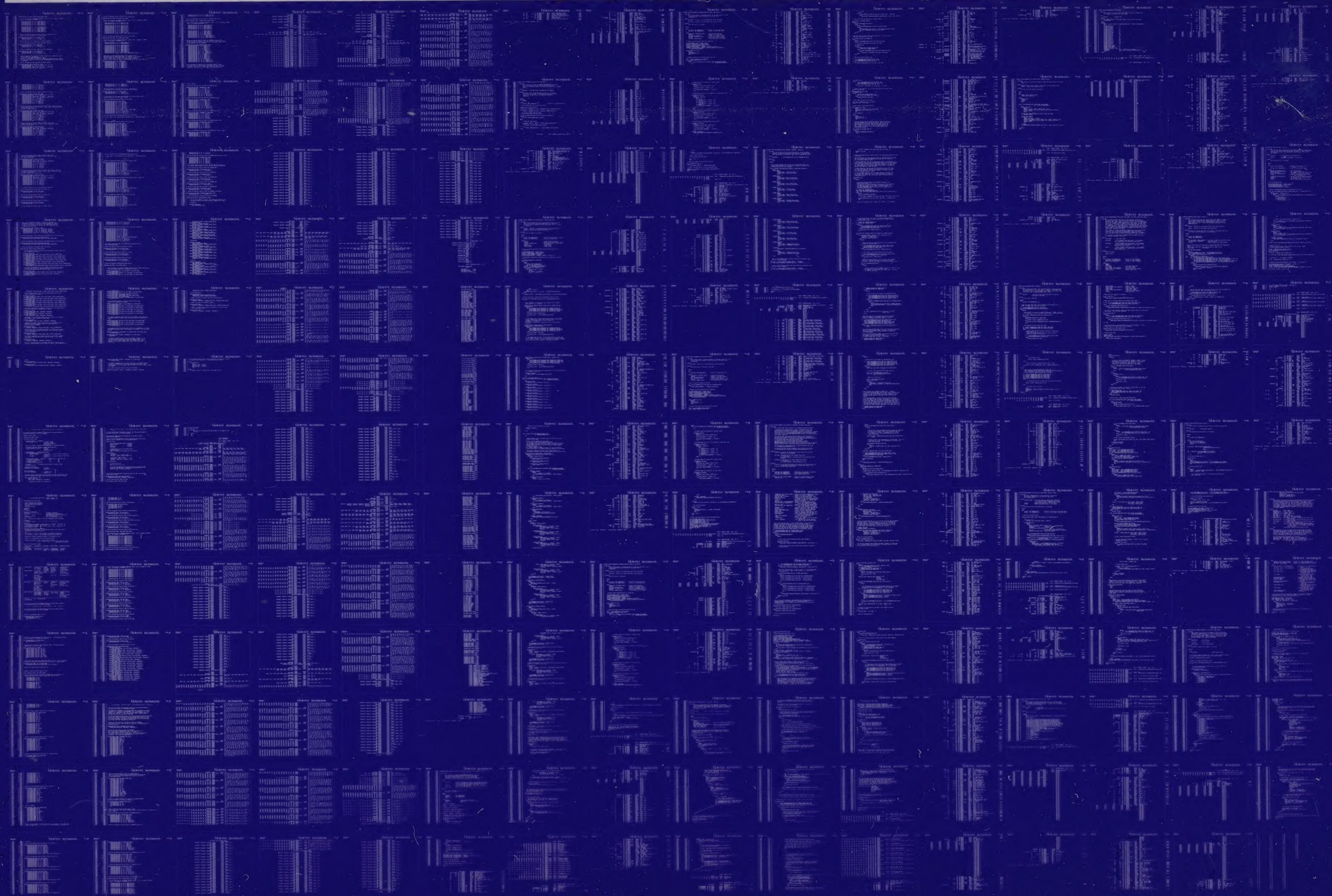
0080 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY



0081 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY



|          |          |          |          |          |          |          |          |          |           |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|
| Frame 1  | Frame 2  | Frame 3  | Frame 4  | Frame 5  | Frame 6  | Frame 7  | Frame 8  | Frame 9  | Frame 10  |
| Frame 11 | Frame 12 | Frame 13 | Frame 14 | Frame 15 | Frame 16 | Frame 17 | Frame 18 | Frame 19 | Frame 20  |
| Frame 21 | Frame 22 | Frame 23 | Frame 24 | Frame 25 | Frame 26 | Frame 27 | Frame 28 | Frame 29 | Frame 30  |
| Frame 31 | Frame 32 | Frame 33 | Frame 34 | Frame 35 | Frame 36 | Frame 37 | Frame 38 | Frame 39 | Frame 40  |
| Frame 41 | Frame 42 | Frame 43 | Frame 44 | Frame 45 | Frame 46 | Frame 47 | Frame 48 | Frame 49 | Frame 50  |
| Frame 51 | Frame 52 | Frame 53 | Frame 54 | Frame 55 | Frame 56 | Frame 57 | Frame 58 | Frame 59 | Frame 60  |
| Frame 61 | Frame 62 | Frame 63 | Frame 64 | Frame 65 | Frame 66 | Frame 67 | Frame 68 | Frame 69 | Frame 70  |
| Frame 71 | Frame 72 | Frame 73 | Frame 74 | Frame 75 | Frame 76 | Frame 77 | Frame 78 | Frame 79 | Frame 80  |
| Frame 81 | Frame 82 | Frame 83 | Frame 84 | Frame 85 | Frame 86 | Frame 87 | Frame 88 | Frame 89 | Frame 90  |
| Frame 91 | Frame 92 | Frame 93 | Frame 94 | Frame 95 | Frame 96 | Frame 97 | Frame 98 | Frame 99 | Frame 100 |

0082

**DIGITAL  
CONFIDE**

EQUIPMENT CORPORATION  
INITIAL AND PROPRIETARY

[illegible]